

**DEEP WATER IMAGE CLASSIFICATION FOR
MARINE USING CONVOLUTIONAL NEURAL
NETWORK**



**STUDENT NAME MUHAMMAD IMRAN
ENROLLMENT NO. 01-249201-006
SUPERVISOR DR. MUHAMMAD MUZAMMAL**

A thesis submitted in fulfilment of the requirements for the award
of degree of Masters of Science (Data Science)

Department of Computer Science
BAHRIA UNIVERSITY ISLAMABAD

MARCH 2022

Approval of Examination

Scholar Name: **MUHAMMAD IMRAN**

Registration Number: **66455**

Enrollment: **01-249201-006**

Program of Study: **MS (DATA SCIENCE)**

Thesis Title: **DEEP WATER IMAGE CLASSIFICATION FOR MARINE
USING CONVOLUTIONAL NEURAL NETWORK**

It is to certify that the above scholar's thesis has been completed to my satisfaction and, to my belief, its standard is appropriate for submission for examination. I have also conducted plagiarism test of this thesis using HEC prescribed software and found similarity index 9 %. that is within the permissible limit set by the HEC for the MS/M.Phil degree thesis. I have also found the thesis in a format recognized by the BU for the MS/M.Phil thesis.

Principal Supervisor Name: **DR. MUHAMMAD MUZAMMAL**

Principal Supervisor Signature:



Date: **03/03/2022**

Author's Declaration

I, MUHAMMAD IMRAN, hereby state that my MS/M.Phil thesis titled is my own work and has not been submitted previously by me for taking any degree from Bahria university or anywhere else in the country/world. At any time if my statement is found to be incorrect even after my graduation, the University has the right to withdraw/cancel my MS/M.Phil degree.

Name of Scholar: MUHAMMAD IMRAN

Date: 03/03/2022

Plagiarism Undertaking

I, solemnly declare that research work presented in the thesis titled **DEEP WATER IMAGE CLASSIFICATION FOR MARINE USING CONVOLUTIONAL NEURAL NETWORK** is solely my research work with no significant contribution from any other person. Small contribution / help wherever taken has been duly acknowledged and that complete thesis has been written by me. I understand the zero tolerance policy of the HEC and Bahria University towards plagiarism. Therefore I as an Author of the above titled thesis declare that no portion of my thesis has been plagiarized and any material used as reference is properly referred / cited.

I undertake that if I am found guilty of any formal plagiarism in the above titled thesis even after award of MS/M.Phil degree, the university reserves the right to withdraw / revoke my MS/M.Phil degree and that HEC and the University has the right to publish my name on the HEC / University website on which names of scholars are placed who submitted plagiarized thesis.

Name of Scholar: MUHAMMAD IMRAN

Date: 03/03/2022

Dedication

I dedicate this work to my family, teachers, and friends who gave me all their love and care. No words can express my gratitude for them, for they have always supported and encouraged me.

Acknowledgements

In preparing this thesis, I was in contact with many people, researchers, academicians, and practitioners. They have contributed to my understanding and thoughts. In particular, I wish to express my sincere appreciation to my main thesis supervisor, Professor Dr. MUHAMMAD MUZAMMAL, for encouragement, guidance, critics, and friendship. I am also very thankful to my co-supervisor Professor Dr. IMRAN JAMAL for his guidance, advice, and motivation. Without his continued support and interest, this thesis would not have been the same as presented here.

Librarians at Bahria University also deserve special thanks for their assistance in supplying the relevant literature. My fellow postgraduate students should also be recognized for their support. My sincere appreciation also extends to all my colleagues and others who have assisted on various occasions. Their views and tips are useful indeed. Unfortunately, it is not possible to list all of them in this limited space. I am grateful to all my family members.

Abstract

Deep learning models perform well in open-air captured image classification problems but, in the case of underwater images, can't find better accuracy due to low-level features. Under-water image classification is a difficult task due to the independent nature of the sea. To overcome low-level issues, we need to implement advanced deep learning models. Meanwhile, the deep learning community focuses on pre-trained deep networks to classify out-of-domain images and transfer learning. This study proposes a model for underwater image classification. We split the Labeled Fishes in the Wild (LFW) dataset into two versions as raw and enhanced before implementation. Strategy II of the transfer learning approach has been used by analyzing the nature of the dataset. We have applied pre-trained CNN architectures such as VGG, ResNet, Xception, and DenseNet on the dataset to find better results. DenseNet121 offers the most accurate predictions for raw and enhanced versions. We achieved 97.84% classification accuracy on the original version and gained 99.35% accuracy on the enhanced version dataset. In the end, we tested our application on Fish Species Image Dataset (FSID) and achieved correct results.

Keywords— Underwater Image Classification, Deep Learning, Convolutional Neural Network, Transfer Learning, Pre-trained CNN Architectures, Image Enhancement.

TABLE OF CONTENTS

AUTHOR’S DECLARATION	ii
PLAGIARISM UNDERTAKING	iii
DEDICATION	iv
ACKNOWLEDGEMENTS	v
ABSTRACT	vi
LIST OF TABLES	x
LIST OF FIGURES	xii
LIST OF SYMBOLS	xiii
1 INTRODUCTION	1
1.1 Background	1
1.2 Research Gap	2
1.3 Problem Statement	3
1.4 Research Questions	4
1.5 Objectives	4
1.6 Significance of the study	5
1.7 Structure of Thesis	5
2 RELATED WORK	6
2.1 Underwater Image Enhancement	6
2.1.1 Enhancement Benchmark Dataset	6
2.1.2 Weak Contrast Image Enhancement - AEIHE	7
2.1.3 Weak Contrast Images - Enhancement	8
2.1.4 Summary: Underwater Image Enhancement	8
2.2 Underwater Image Classification	8
2.2.1 Low-resolution Image - Fish Recognition	8

2.2.2	Fish Image Generation and Classification	9
2.2.3	Fish Species Classification	9
2.2.4	ResFeats: Underwater image classification	10
2.2.5	Underwater Image Classification using ML	10
2.2.6	Underwater Image Classification using DL	11
2.2.7	Underwater Classification using Monocular Image	11
2.2.8	Multi-label Underwater Image Classification	11
2.2.9	DUICM: Deep Underwater Image Classification Model	12
2.2.10	Underwater Mine Classification using Transfer Learning	13
2.2.11	Summary: Underwater Image Classification	13
2.3	Dataset: Labeled Fishes in the Wild	13
2.3.1	Labeled Fishes in the Wild	13
2.3.2	Real-Time Fish Recognition using CNN	14
2.3.3	Summary: Labeled Fishes in the Wild	14
3	RESEARCH METHODOLOGY	15
3.1	Current Method	15
3.2	Proposed Methodology	16
3.3	Dataset	17
3.4	Image Enhancement - Adaptive Histogram Equalization	19
3.4.1	Contrast Limited Adaptive Histogram Equalization - CLAHE	19
3.5	Convolutional Neural Network	19
3.5.1	Convolutional Layer	20
3.5.2	Pooling Layer	21
3.5.3	Fully Connected Layer	22
3.6	Transfer Learning	22
3.7	CNN Architectures	24
3.7.1	VGGNet	24
3.7.2	ResNet	24
3.7.3	DenseNet	25
3.7.4	Xception	27
3.7.5	CNN Architectures: an overview	27
3.8	Evaluation Measurement	28
3.8.1	Classification Problem and Label Encoding	28
3.8.2	Activation Function	29
3.8.3	Loss Function	30
3.8.4	Confusion Matrix	31
3.8.5	Accuracy Metrics	32
4	DATA ANALYSIS, RESULTS & FINDINGS	33

4.1	Experimental Setup and Dataset	33
4.2	VGG16 as a Model	36
4.3	VGG19 as a Model	37
4.4	ResNet50 as a Model	38
4.5	Xception71 as a Model	39
4.6	DenseNet121 as a Model	40
4.7	Results Comparison	41
4.7.1	Best Model - Raw/Original Images	42
4.7.2	Best Model - Enhanced Images	42
4.7.3	Samples Predictions	44
4.7.4	Model Summary	45
4.8	Application Results	47
5	DISCUSSION AND CONCLUSION	49
	REFERENCES	52
	APPENDICES A-Y	61

LIST OF TABLE

1.1	Research questions for the proposed model	4
3.1	Performance Evaluation (RUI)	16
3.2	Performance Evaluation (EUI)	17
3.3	Comparison between ReLu and Softmax activation functions	17
3.4	When to use which strategy of pre-trained networks	23
3.5	A typical architectures in convolutional neural networks	28
4.1	Most used tools and technologies in the proposed research	34
4.2	The class balance of the dataset	34
4.3	Data Preprocessing	35
4.4	The constant parameters	36
4.5	VGG16 on raw version	37
4.6	VGG16 on enhanced version	37
4.7	VGG19 on raw version	38
4.8	VGG19 on enhanced version	38
4.9	ResNet50 on raw version	39
4.10	ResNet50 on enhanced version	39
4.11	Xception71 on raw version	40
4.12	Xception71 on enhanced version	40
4.13	DenseNet121 on raw version	41
4.14	DenseNet121 on enhanced version	41
4.15	Raw Version of FITW Dataset	42
4.16	Model-5 (Classification Report - Raw version)	43
4.17	Enhanced Version of FITW Dataset	45
4.18	Model-5 (Classification Report - Enhanced version)	45
4.19	Ten samples predictions using best models	47
4.20	Model Summary	47
4.21	Application Results	48
5.1	Comparison of proposed method with existing works (Raw/Original Images)	49

5.2 Comparison of proposed method with existing works (Enhanced Images) 50

LIST OF FIGURE

3.1	CNN framework of DUICM model.	16
3.2	The proposed methodology for underwater image classification.	18
3.3	Workflow of a classical CNN.	20
3.4	Workflow of a convolutional layer.	21
3.5	The pooling layer, also known as sub-sampling layer.	21
3.6	The fully-connected layer of a classical CNN.	22
3.7	The concept of with and without transfer learning.	23
3.8	A visualization of VGGNet architecture.	25
3.9	(a) Plain layer (b) Residual Block [1]	26
3.10	The standard structure of a Dense block with three layers.	26
3.11	The structure of Xception architecture and dept-wise separable convolutions.	27
3.12	Confusion matrix or the table of confusion.	31
4.1	An analysis of fish species.	35
4.2	Before and after applying the CLAHE method.	35
4.3	Model 5 (accuracy w.r.t raw version)	43
4.4	Model 5 (loss w.r.t raw version)	44
4.5	Model 5 - (Confusion Matrix w.r.t raw version)	44
4.6	Model 5 (accuracy w.r.t enhanced version)	45
4.7	Model 5 (loss w.r.t enhanced version)	46
4.8	Model 5 - (Confusion Matrix w.r.t enhanced version)	46

LIST OF SYMBOLS

- \mathcal{X} – Variable
- exp – Exponential
- σ – Sigma
- Σ – Sum

CHAPTER 1

INTRODUCTION

1.1 Background

Computer vision is a broad field in deep learning which handles object detection, object tracking, image segmentation, and image classification. Computer vision is a platform for classifying images in different conditions such as underwater and open air. Deep learning algorithms were applied and gained success on many large datasets for open-air image classification but make overfitting when applied these algorithms on small datasets [2]. In open-air captured image classification problems, the deep learning algorithms operate effectively, but, in the case of underwater images, the deep learning models didn't find better accuracy due to low-level features. Under-water image classification is a difficult task because of the autonomous nature of the ocean [3]. The classification algorithms for underwater images need to address additional issues that arise from the loss of quality of an image due to light scattering, absorption, and reduction of image saturation which intensely affect the visual perception [4]. To overcome low-level issues, we need to implement advanced deep learning models. Preprocessing is necessary for these problems to improve the image colors and advance image classification [5]. Suitable image processing algorithms play a vital role in enhancing the vision of underwater images. Convolutional Neural Network (CNN) is one of the most promising classifiers for recognizing different kinds of images. Extracting features from an image is the main focus of classification and is carried out by grouping pixels into various labels [6]. Supervised and unsupervised learning are the main types of machine learning used in classification problems. Supervised learning is an algorithm that extracts features, train the model and analyze by using labeled data. However, unsupervised learning also plays a vital role in image classification in which the pixels of the image are automatically bunched into clusters [7]. Currently, deep learning models are broadly used in image processing to get accurate predictions. The reason behind it is its fast-computing performance to handle and execute large datasets by learning things from dept [8]. Generally, CNN is an appropriate classification technique while performing image

processing tasks. CNN can automatically recognize the main features of the image without any human supervision [9]. Deep learning algorithms deal with the multilevel representation of data to make input images clear and understandable such as from lower-level to higher-level. ImageNet is a large dataset available in deep learning, has included manually-annotated data that support test algorithms to organize multimedia data [10]. For classifying images of large datasets, choosing CNN is the foremost procedure in deep learning. Like other deep learning models, the CNN's uses Stochastic Gradient Descent (SGD) to train a different model such as graphical models (where the amount of data is large). CNN has various training layers that are helpful in low-resolution image classification. For example, the input layer, convolutional filters layer, pooling layer, fully-connected layer, and the output layer [11, 12]. The ability of parameter tuning is a valuable technique to advance CNNs for image classification [13]. Many data sets are available for the traditional underwater image classification, but we are trying to figure out a more effective method for deepwater image classification.

1.2 Research Gap

In this study [14], we have found that the current models follow inaccurate algorithms that implicitly limit the progress of underwater image enhancement. And, some users did not understand the physical model of underwater images. So, they might ignore the presence of the backscatter effect in long distances. The proposed methodology in [15] can only improve contrast in extreme conditions. The results in [16] argue that small images may lose too much information when enlarging images or transforming to gray-scale. The researcher in [17] can achieve better results by applying more CNN architectures with more fake images. Due to the effect of background noise and other underwater stuff, the proposed algorithm [18] couldn't have achieved 100% accuracy as some pictures couldn't predict correctly. The proposed system can improve by further applying contrast enhancement techniques to overcome the lost features problem. The ResFeats [19] method might not achieve better classification accuracy due to the complex nature of underwater images and a wide range of classes, such as benthic and fish, etc. Executing different deep learning algorithms such as convolutional neural networks may increase the classification performance of the proposed method [20]. Deep et al. [21] might increase the accuracy by preprocessing and augmenting the dataset and applying other deep learning methodologies such as transfer learning to enhance the performance of the proposed method. The dataset collected in this study [22] is from the same water condition doesn't allow us to evaluate the model in different water conditions. The label assigned to a benthic image was not the main idea from the dataset. So, this created several issues but also revealed interesting areas in multi-label image classification

[23]. The model [24] can be more effective by adding more layers and applying them to large datasets. Transferring high-frequency SAS-image CNNs to low-frequency SAS images has not been done yet. Also, the relative benefit of transfer learning as a function of training dataset size requires a larger dataset than the existing one [25].

1.3 Problem Statement

Nowadays, computer vision is the top trending technology and the rapidly increasing research in image classification and object detection towards open-air captured images. The existing systems are mostly attentive to light conditional approaches of image classification. In the case of underwater image classification, many challenges occur, such as:

- Blur
- Distinct nature
- Light scattering
- Absorption
- Saturation
- Low-light illumination problem

The underwater captured images might be a blur, distinct nature due to different types of waters like normal and shallow water, and light scattering because of displacement of the light direction. Similarly, another problem is the size reduction at the center of deepwater images due to the blueish color under the water. This problem is also called absorption. The further challenge that occurs in the underwater taken image is saturation. Saturating the images refers to the decrease of the intensity of the image corners. Correspondingly, the main issue in underwater captured images is that they might be dark because of the low-light illumination problem. By observing these problems and challenges, there will be a requirement for a precise deep learning algorithm for underwater image classification, which can perform better than existing systems.

1.4 Research Questions

Table 1.1: Research questions for the proposed model

S.No	Question	Description
1	How to overcome the challenges of underwater image classification?	To choose the most used and suggested deep learning methods applied in underwater image classification.
2	How will the accuracy of the raw underwater images be increased?	To recognize the effective classification methods applied with original/raw underwater images for classification.
3	How will the accuracy of the enhanced underwater images be increased?	To identify the efficient classification and latest image enhancement techniques for underwater image classification.
4	Which CNN model predicts more precisely for underwater images?	Analysis and comparison of CNN architectures applied on original and enhanced dataset versions.

1.5 Objectives

The main purposes of deep-water image classification are to:

- Train the proposed base model with more layers.
- Train with large underwater image dataset using faster GPUs to gain more accurate results.

This research aims to use the CNN algorithms to predict the underwater images and achieve better accuracy. As shown in the literature review section, several methods had used for image classification. Almost all of them used CNN or CNNs sub-techniques like ResNet etc. In this study, we are applying CNNs on a large dataset than the turbid underwater image dataset [26] (TURBID Dataset) as mentioned in the future work for classifying underwater images in the base paper [24]. In this research, we are using Labeled Fishes in the Wild dataset [27]. We are implementing different CNN architectures by analyzing several works from the literature review. So, we will identify which model gives us better accuracy that we compare the evaluation with the results of our base model(DUICM) and other existing works based on underwater image classification.

1.6 Significance of the study

We are implementing classification methods on the chosen features from Labeled Fishes in the Wild Dataset. There are almost 1400+ images taken from different videos in the underwater dataset. Our proposed method aims to overcome low-level feature issues using a large dataset instead of a small dataset increased performance in both forms, such as original and enhanced. And applying the pre-trained models that are much more suitable methods than traditional ones to go deeper with layers. It may also save time during model creation by using a transfer-learning approach instead of making deep models from the beginning. Alternatively, our proposed model will be helpful for many purposes, such as mine classification, for military purposes, finding dead bodies beneath the water, finding treasure, and many more.

1.7 Structure of Thesis

The rest of the paper has divided as follows: Section 2 describes the review of different related works. Section 3 explains the proposed methodology and briefly discusses various techniques used in this work. In Section 4, we discuss the experiments and results. Section 5 presents the conclusion and future research directions.

CHAPTER 2

RELATED WORK

Classification models perform well in the case of open-air captured images and classify with better accuracy. Underwater captured images have low-level features, and the system needs well-designed algorithms that perform higher and complex calculations for the underwater taken image. We have been discussing various methodologies for image enhancement, different techniques, and traditionally used approaches for deepwater image classification.

2.1 Underwater Image Enhancement

2.1.1 Enhancement Benchmark Dataset

The aim of the study [14] was to analyze underwater image enhancement using extensive tainted images. Researchers generated a benchmark dataset for classification called the UIEBD dataset that has 950 underwater captured images. The collected images in the dataset from multiple light effects such as natural, artificial, or a mixture of both lights. The UIEB dataset provides a platform to evaluate the performance of different deep-water enhancement models. Furthermore, by using the generated UIEB dataset, scientists conducted a comprehensive survey of single deep-water image enhancement techniques ranging from qualitative to quantitative estimations. Researchers proposed a CNN algorithm named Water-Net trained by the generated UIEB dataset for underwater image enhancement, which determines the evaluation of UIEB and the advantages of the proposed network. Also, the proposed CNN model enhances the advancement of computer vision-based underwater image enhancement. Detailed experimentation of underwater image enhancement had done using the generated UIEB dataset. The applied image enhancement methods were both qualitative and quantitative such as UDCP [28], Retinex-based [29], blurriness-based [30], Red Channel [31], two-step-based [32], fusion-based [33], regression-based [34], histogram prior [35] and GDCP [36]. These experiments demonstrate the efficiency of the proposed Water-Net and also specify the generated dataset can be useful for training CNNs. The results of the proposed Water-Net compared with

traditional CNNs: considering MSE, it gave the lowest from other CNNs with the value of 0.7976, in the PSNR metric it had the highest value of 19.1130 and, the SSIM also had the highest of 0.7971, respectively. On the other hand, Water-Net had 2.57 of the highest average score and 0.728 of the lowest standard deviation amongst all CNNs.

Research Gap: By observing this study, we have found two research gaps, such as the current models following inaccurate image generation algorithms that implicitly limit the progress of underwater image enhancement. And, some users did not understand the physical model of underwater images. So, they might ignore the presence of the backscatter effect in long distances. Using inaccurate image recognition algorithms keeps image classification and other computer vision areas standstill.

2.1.2 Weak Contrast Image Enhancement - AEIHE

Many photo capturing devices have developed in the last decades, such as digital, mobile, and security cameras. And these cameras are usually used by humans for personal use. Contrast improvement is the main factor of computer vision used in various domains such as healthcare, military and satellite images, etc. Lack of image enhancement laid to illumination, limited user experience, and poor-quality devices. Histogram equalization (HE) has been currently used to tackle such issues. It's an image enhancement technique, which improves the quality of the image. Adaptive entropy index histogram equalization (AEIHE) system initiated to enhance the quality of an image [37]. Initially, AEIHE splits the image into three sub-parts to increase its local feature. Each sub-part of the image used a different area. The clip limit depends on the richness of their feature and their design. Then, a new parameter named Entropy-Index has applied to confirm the highly-rich areas of the sub-part while preserving its design. AEIHE was measured to be a brilliant HE-based method for image enhancement. AEIHE was evaluated based on the results of qualitative and quantitative analysis. AEIHE was successfully formed good quality images with promising enhancements, information details, and richness, and superbly preserved design with fewer effects from noise, undesirable objects, and over-enhancement.

Research Gap: Non-real-time captured images such as healthcare and the personal camera do not depend on time while enhancing images. But focus on the making of the best results. Although, current techniques for real-time enhancement applications can quickly generate results. In many cases, these techniques have been unable to produce optimal results and demonstrate poor performance. Whereas methods for non-real-time applications may take high enhancing time yet can generate the best results in some cases. AEIHE is designed for non-real-time applications and focuses upon gaining accurate resultant images that are real and pleasant to the human eye

with contrasted local details, fewer noise effects, and preserved structure.

2.1.3 Weak Contrast Images - Enhancement

A modified CNN architecture was proposed in [15] to achieve improvement in poor contrast images. DND, SIDD, and RENOIR are the three publicly available datasets used for noise reduction. Then, the authors observed pixel distribution attributes of the RGB channels of poor contrast images. And, the histogram distribution was given, which had used to fix the size of the kernels of the CNN. Afterward, a multilayer convolutional neural network was created and fed the original image as input. The model gave a noisy image output. The results demonstrate that the proposed algorithm achieved a higher peak signal-to-noise ratio (PSNR) and an advanced structural similarity index (SSIM) than other image contrast approaches.

Research Gap: However, we have found that the proposed methodology can only improve contrast in extreme conditions.

2.1.4 Summary: Underwater Image Enhancement

Note that the performance of image enhancement methods differs from one field to another heavily depending on its objective. Some existing works used traditional image enhancement techniques that imposed many challenges such that can only improve contrast in extreme conditions [15], and avoid the existence of the backscatter effect in long distances [14]. However, our experiments show that the enhancement technology increases the vision of images for the human eye, removing noise and blurring, improving contrast, and revealing details. We use the Contrast Limited Adaptive Histogram Equalization (CLAHE) [38] for image enhancement as a method.

2.2 Underwater Image Classification

2.2.1 Low-resolution Image - Fish Recognition

In [16] the researchers proposed a model that predicts fish species from low-resolution images. The features extraction wasn't simple from fuzzy images. Advanced deep learning technologies had used to extract discriminative features from underwater captured images such as Network in Network (NIN) [39] and PCANet [40]. After that, they trained a linear SVM model to classify different fish species. The researchers trained the linear SVM classifier individually for each deep learning model using libSVM library [41]. The linear kernel had used to train the model since the task is not a binary classification problem. Experiments carried out on Fish-OR and Fish-SR datasets. The researchers take traditional dense sift (Dsift-Fisher) features [42] and Gabor features [43] to predict using the image of the FishCLEF2015

dataset as comparison methods. Then, the SVM model had trained using both features. The deep methods achieved much higher performance than the traditional ones. The results show that the proposed methodology gained accurate outputs for fish classification.

Research Gap: The results argue that small images may lose too much information when expanding to large images or converting to gray-scale.

2.2.2 Fish Image Generation and Classification

Scientific studies on fish species have a crucial role in underwater life. Fish images are first collected by different vehicle systems and then interpreted manually by researchers. The field of computer vision has enhanced to a great extent due to the rise of deep learning algorithms. Underwater image classification is a difficult task than open-air classification. Usually, traditional CNN needs big data with high features to attain better evaluations since fish images have low features and small datasets available. In [17], authors prepared three types of fish datasets that expand the training data to assure the quality and quantity. For each of the few-shot classes, a translation model had been used to generate high-quality images. It achieved better performance in fish image classification. The results show that adding more fake images to training data may increase higher-classification accuracy. VGG16 and ResNet50 were used to achieve better performance.

Research Gap: The researcher can achieve better results by applying more CNN architectures with more fake images.

2.2.3 Fish Species Classification

This research [18] attempted to recognize underwater images of fish species using image classification, deep learning, and convolutional neural networks. The dataset used for the proposed methodology was Fish4Knowledge [44] of 27,142 images. In the first step, the noise had removed from the dataset. Then, a computer-vision-based tool had applied for data preprocessing. The preprocessing tool had used to remove the underwater dirt, non-fish objects, and obstacles in the images. Preprocessing had done to enhance feature recognition and training of the CNN model. After that, classification for fish labels had done using deep learning algorithms. The purpose of training an algorithm was to tune a network that minimizes the error rate between the actual and predicted outputs. Max pooling layer is used for well-separated frames to divide the input image. Different activation functions were applied and compared for getting a better evaluation, such as ReLU, Tanh, and Sigmoid. The proposed method of classification of fishes gave 96.29% of accuracy that was the highest compared to current and other traditional approaches used for this application.

Research Gap: Due to the effect of background noise and other underwater stuff, the proposed algorithm couldn't achieve 100% accuracy as some pictures couldn't predict correctly. The proposed system can improve by further implementing image enhancement techniques to tackle the lost features issue.

2.2.4 ResFeats: Underwater image classification

Underwater images depend on an advanced image capturing devices. Most of the photos captured by these systems do not get annotated due to fewer resources. So, the researchers' data wasn't much to train a deep learning model. Currently, the deep learning community mainly focuses on implementing pre-trained techniques and transfer learning to classify images. In [19] ResFeats (image features) proposed to measure how accurately the features had extracted from the different convolutional layers of a pre-trained deep residual network on ImageNet. The researchers combined ResFeats extracted from various layers to attain favorable deep features. Also, it had proved that ResFeats gradually performed better than other CNN counterparts. The experimental results show the performance of ResFeats with improved classification accuracies on RSMAS, Benthos15, MLC, and EILAT datasets.

Research Gap: The ResFeats method may not achieve better classification accuracy due to the complex nature of underwater images and a wide range of classes, such as benthic and fish.

2.2.5 Underwater Image Classification using ML

Over the last few years, an underwater study has increased significantly. Presently, Side Scan Sonar (S3), Remotely Operated Vehicle (ROV) and, others are existing data collection devices used in underwater research. These devices provide data about the ocean surface, objects, and species. So, choosing an effective and appropriate feature is a difficult task. It is hard to classify underwater images because of the small number of datasets. Also, it's a difficult task due to low light intensity. Machine learning-based model had applied in this paper [20], called the Bag of Features model to tackle the mini dataset issue. The dataset images of seven classes had collected from shallow water using ROV. Speeded-Up Robust Features (SURF) algorithm had used to extract the features from the dataset. Machine learning algorithms such as Support Vector Machines (SVM) were used as a classifier in the Bag of Features model to maximize accuracy and obtained 93% from the experiments. Finally, this model gives better performance for the underwater image dataset.

Research Gap: Executing different deep learning algorithms such as convolutional neural networks may increase the classification performance of the proposed method.

2.2.6 Underwater Image Classification using DL

Many computer-vision-based technologies had introduced to predict fish labels accurately. In [21], DeepCNN, DeepCNN-SVM, and DeepCNN-KNN frameworks proposed that were performed better than current methods for underwater fish classification in terms of accuracy, precision, recall, and f1-score. The proposed method DeepCNN-KNN achieved an accuracy of 98.79%.

Research Gap: There is also a chance to increase the accuracy by preprocessing and augmenting the dataset and applying other deep learning methodologies such as transfer learning to enhance the performance of the proposed method.

2.2.7 Underwater Classification using Monocular Image

Underwater captured images become the main issue for robotic applications due to the high intensity of absorption. The main problem requires a depth map for various applications. Lack of large datasets is one of the main issues to train and validate the model. Some methods suggested for result measuring depends on the physical approach and deep learning model. By analyzing the advantages and disadvantages of each type of model, the objective of [22] is to obtain an efficient depth map for input RGB images. The proposed work provided a wide range of scenarios for underwater captured images. The authors adapted the model-based advanced technique to raw images to measure depth maps from compressed images. The learning-based method shows better results in both cases. So, the proposed framework correctly predicts the water type of the case images and correctly selects the best technique to measure the depth map.

Research Gap: The dataset collected in this study is from the same water condition doesn't allow us to evaluate the model in different water conditions.

2.2.8 Multi-label Underwater Image Classification

In this study [23], researchers trained a multi-label image classification model on the underwater image datasets using ResNet50, which is a pre-trained CNN architecture. In the first stage, the dataset only provided single-label images. The data set had split into subsets such as 80% used for training and 20% was used for validation and testing. The performance reported on the test data. Additionally, the model ran on video transects data to evaluate the measurement of the models to operate in a functional environment. The collected transects weren't well-adjusted amongst classes. They delivered different scenes of the dataset, especially for the midwater data. The proposed model trained for 15 species and clustered depending on the number of pictures for each class. And the amount of almost 1,000 images per class. The dataset had generated in the form of 33,064 sets of images. This

model had trained for a single-label classification problem and has been able to classify multi-label classification problems using a Top-N scoring method. The relative accuracy and unrecognized classes had proved as part of this method. By applying the above-stated scoring method, the researchers found Top 1 accuracy of 85.7% and Top 3 accuracy of 92.9% on the FathomNet test data. Then the performance of the algorithm was tested using a midwater transected dataset. The FathomNet training set had a combination of many focused and zoomed-in images. These images were enhanced using a limited spatial resolution of objects in the midwater transect footage. So, the proposed algorithm performed poorly due to differences between the iconic imagery (e.g., FathomNet or benthic training data) and differences in resolution and scale of labels (e.g., midwater transect data). Another experiment had done by training a model of 15 benthic classes out of 17. Firstly, the researchers limited the number of training images to 700. Then, removed this threshold to include the classes available in the midwater transect dataset. This experiment resulted in a dataset with a total of 33,064 images. The authors found Top 1 accuracy of 72.4% and Top 3 accuracy of 92.8% on the test data. A quickly recognizable difference between the midwater and benthic outcome was the comparisons between Top 1 and Top 3 accuracy metrics. There was a noticeable increment in the benthic dataset moving to Top 3 accuracy due to the multi-concept nature of deep-water taken images in FathomNet. While midwater dataset images had considered being commonly single-concept.

Research Gap: The label assigned to a benthic image was not the main idea from the dataset. So, this created several issues but also revealed interesting areas in multi-label image classification.

2.2.9 DUICM: Deep Underwater Image Classification Model

The aim of the deep underwater image classification model [24] (DUICM) was to apply a convolutional neural network (CNN) on underwater images. It helped train and classify the turbid images for the chosen features from the Benchmark Turbid Image Dataset. The dataset had further converted into two more versions, raw and enhanced image datasets. The accuracy had checked concerning activation functions, no. of epochs, and confusion matrix. The DUICM gave a 97% accuracy rate on the enhanced version and offered 69% on the raw version dataset. The accuracies show for ReLU as an activation function.

Research Gap: The model can be more effective by adding more layers and applying them to large datasets.

2.2.10 Underwater Mine Classification using Transfer Learning

CNN's designed and trained for an underwater mine classification using SAS data in [25]. This work determined the efficiency of using CNNs for UXO recognition when SAS had fewer training data. Transfer learning allowed better classification results compared to training from scratch. These discoveries can be assured to minimize training data requirements in a broad range of remote-sensing issues characterized by limited data.

Research Gap: By reviewing, we found that transferring high-frequency SAS-image CNNs to low-frequency SAS images has not been done yet. Also, the relative benefit of transfer learning as a function of training dataset size requires a larger dataset than the existing one.

2.2.11 Summary: Underwater Image Classification

As compared with the existing work, our proposed method has the following advantages:

- The dataset used in this study contains images from different water conditions such as shallow, mid, and deep-water, which allow us to evaluate the model in, unlike water conditions.
- Implementing an image enhancement method such as CLAHE tackles the lost features issue.
- The proposed model performs more effectively by adding more layers (using pre-trained CNN architectures) and applying them to a large dataset.
- The model achieves more precise predictions on both the enhanced and the original versions.

2.3 Dataset: Labeled Fishes in the Wild

2.3.1 Labeled Fishes in the Wild

A new underwater image dataset, “labeled fishes in the wild,” was collected from ROV survey imagery. The dataset has been made available for object detection, fish species classification, and fish tracking. The dataset had developed using images of *Sebastes* and other species captured from ROV cameras during rockfish surveys, mainly in the southern California Bight between 2000 and 2012. Automation of fish recognition benefits NOAA by lessening the number of personnel hours needed for video surveys [45].

Research Gap: Recognition of fish species and tracking of detected fish to automate counting had been missing from the proposed method.

2.3.2 Real-Time Fish Recognition using CNN

The CNN-based real-time fish recognition method proposed in this paper [46]. The YOLO detector was acquired for fish recognition and trained the network using a custom dataset named Fishes in the Wild. The model offered 93% classification accuracy, 0.634 IOU between the actual and predicted bounding boxes, and 16.7 frames per second of fish recognition. The proposed network outperformed the HOG models showed a much faster processing speed. It indicated that neural networks efficiently classify low-resolution, noisy, and blurred underwater images.

Research Gap: The model predicts all fish as a ‘positive’ class instead of fish species. Annotating images by species can be helpful to classify, observe, and detect fish for certain species.

2.3.3 Summary: Labeled Fishes in the Wild

The labeled fishes in the wild dataset had applied in object detection and other computer vision tasks. In our experiments, for the first time, we have done image classification on the fish species using labeled fishes in the wild dataset.

CHAPTER 3

RESEARCH METHODOLOGY

This chapter discusses the implementation details of approach. Initially we discuss the challenges faced for which we employ proposed approach followed by the details associated with intended framework.

3.1 Current Method

The existing system has based on multi-label underwater taken images. The CNN structure of the DUICM model includes five layers:

–**Layer 1:** Input image of size 64x64, every image size generating 32 features map of size 64x64 and 32 number of filters.

–**Layer 2:** Using 64 number of filters and image size of 5x5 generating 64 features map of size 26x26.

–**Layer 3:** Applied ReLU activation function in this layer, which holds 96x96x32 size of the input image.

–**Layer 4:** It is a max-pooling layer using a pooling window of 2x2. Which reduces spatial dimensions 10x10.

–**Layer 5:** The last layer has 128 neurons and is a fully connected layer. Softmax activation/transfer function applied in this layer.

The complete workflow of the current model (DUICM) using CNN as shown in Fig. 3.1.

The current model uses CNN for underwater image classification using turbid dataset. As shown in the Fig. 3.1 that how the model works. The model is evaluated with two types of images: Raw underwater taken images and Enhanced underwater images. Performance measures of all the 5 epochs are shown in Table 3.1 for Raw Underwater Images.

Performance measures of all the 5 epochs are shown in Table 3.2 for Enhanced Underwater Images.

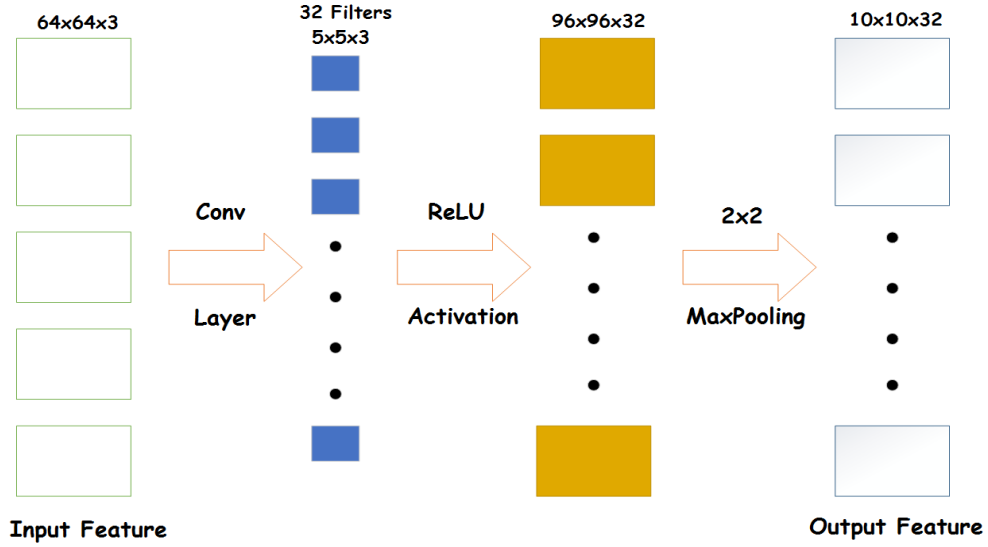


Figure 3.1: CNN framework of DUICM model.

Table 3.1: Performance Evaluation (RUI)

S.No	Epoch	Loss	Accuracy
1	1/5	0.8156	0.5021
2	2/5	0.6978	0.5189
3	3/5	0.6321	0.6312
4	4/5	0.5629	0.6578
5	5/5	0.3215	0.6938

The results show the better accuracy by comparing Raw and Enhanced versions, where the enhanced images have better accuracy of 97.37% than Raw Underwater Images (RUI). Table 3.3 shows the comparison of activation functions that have with better accuracy. For both Raw and Enhanced Underwater Images, ReLu activation function gives better accuracy as compare to Softmax.

3.2 Proposed Methodology

The accuracy of enhanced underwater images is 97.37%, and for raw images, the accuracy is 69.38% in the current work. However, the main focus of our proposed work is to increase the accuracy of the original version without losing the enhanced version's accuracy. In the proposed model, we have been applying CNN architectures such as VGG, ResNet, Xception, and DenseNet to find better results. The current CNN model only has five layers. It has been mentioned in the future

Table 3.2: Performance Evaluation (EUI)

S.No	Epoch	Loss	Accuracy
1	1/5	0.2255	0.7966
2	2/5	0.1365	0.8397
3	3/5	0.0356	0.8963
4	4/5	0.0468	0.9541
5	5/5	0.0723	0.9737

Table 3.3: Comparison between ReLu and Softmax activation functions

Activation	Normal Image Accuracy	Enhanced Image Accuracy
ReLU	69.38%	97.37%
Softmax	54.01%	86.23%

work of base paper to test with larger layers models. Therefore, we are using pre-trained models that consist of 16 to 121 layers. It has also been mentioned in future work to apply models on larger datasets. For this purpose, we are using a larger dataset named Labeled Fishes in the Wild (LFITW) that has 1400+ underwater taken images instead of the Benchmark Turbid Image Dataset, which has only 82 underwater captured photos. The workflow of our proposed model has shown in Fig. 3.2.

Our proposed system is composed of various layers. The dataset preparation to results analysis is performed at these layers, based on different architectures. These layers include data preprocessing and augmentation, implementation of CNN architectures, development of application using trained weights, and the results analysis and comparison. The description of these layers and the proposed architecture has provided in the following subsections. Accuracy is the evaluation measure used for result comparison.

3.3 Dataset

The Benchmark Turbid Image Dataset in the existing model is very small, consisting of 82 turbid images. The images are both in a raw and enhanced form. So, in this paper, we will be using Labeled Fishes in the Wild Dataset that contains 1475 annotated frames from different videos. The purpose of using the dataset is

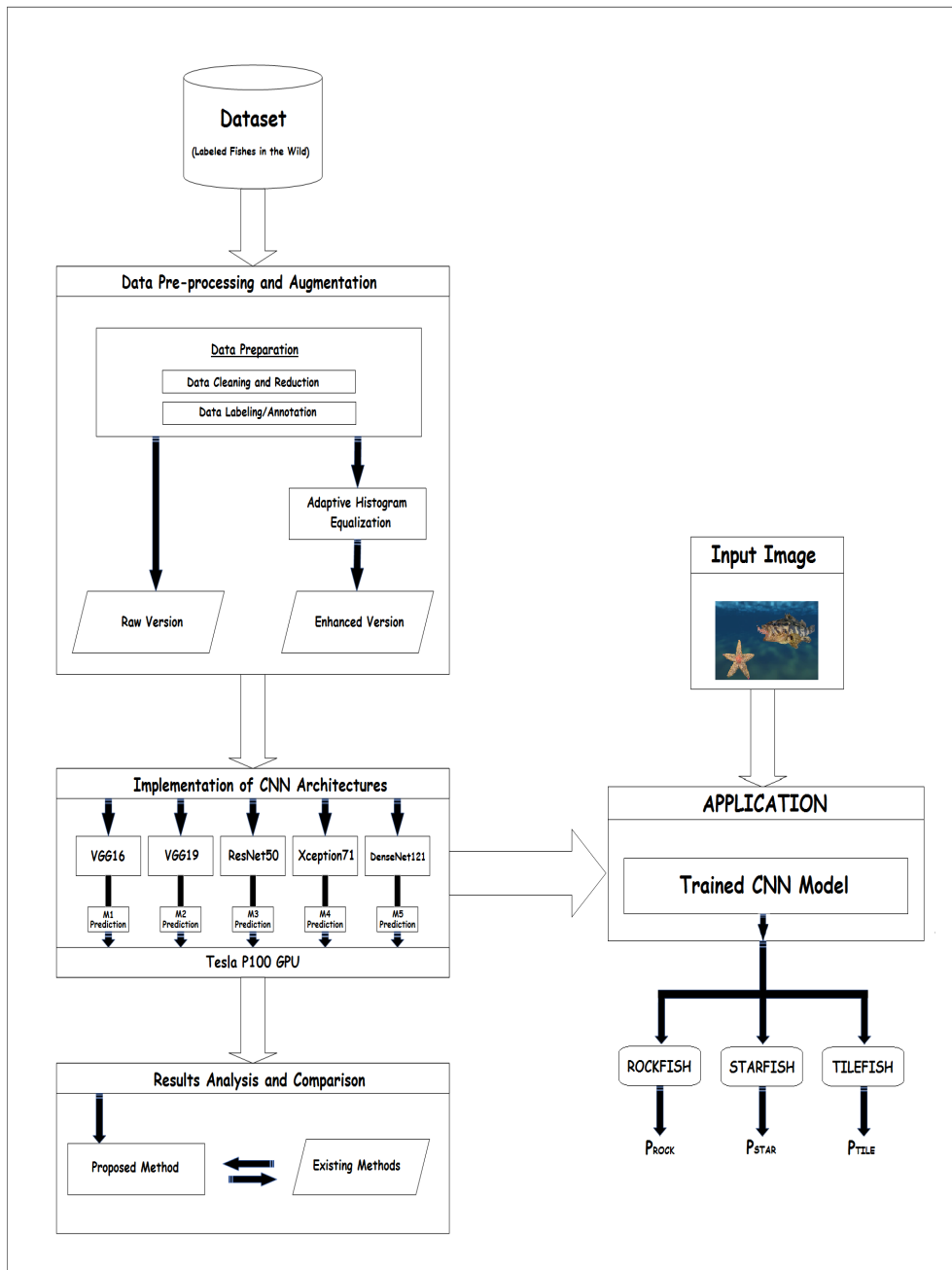


Figure 3.2: The proposed methodology for underwater image classification.

that we require a large dataset for underwater classification, and there are fewer available datasets. We selected the dataset with 1475 frames from several videos. The proposed method is only limited to underwater taken images and deep-water classification. There are a lot of datasets available for open-air image classification,

but the work done in the underwater world is much lesser than open-air imagery data. We are using the Labeled Fishes in the Wild [27] dataset in our proposed method. The dataset consists of images of fish, invertebrates, and the seabed. The images were collected using a camera-operated remotely operated vehicle (ROV) for fisheries researches. This dataset provides fish and other sea animal detection, multiple animal target tracking, evaluation of aquatic animals in stereo image pairs, and fish species classification. Another point to consider is that we need a dataset for testing the application. For this cause, we have chosen images from Fish Species Image Data (FSID) [47]. The FSID contains approximately 4000 images from 468 species of fish.

3.4 Image Enhancement - Adaptive Histogram Equalization

Adaptive histogram equalization (AHE) is a decent image enhancement technique for natural, underwater, dark, blurred, and other images. The automatic process and efficient performance of all contrast presented in the image features make it eligible for the proficient image enhancement technique. AHE has the advantages of being static and reproducible and requiring the examiner to observe only a single case image. The basic Histogram Equalization (HE) enhancement has based on the pixels in an area surrounding its contextual location. But the simple HE method is too slow, and the contrasted image has undesirable features under certain circumstances. AHE increases its speed on various processors and splits the image into different tiles, then implements HE into these tiles [48, 49, 50].

3.4.1 Contrast Limited Adaptive Histogram Equalization - CLAHE

CLAHE is a form of AHE in which the contrast enhancement is limited to control the noise amplification problem. CLAHE is an algorithm for local contrast improvement that uses histograms processed over various tile areas of the image [38]. The CLAHE model has three main parts: tile creation, histogram equalization, and bilinear interpolation. In the first step, the case image has divided into different tiles. Then histogram equalization is then processed on each tile using a pre-defined clip limit. In the end, the tiles have joined together using bilinear interpolation to create an improved contrasted image [50, 51].

3.5 Convolutional Neural Network

Convolutional Neural Network (CNN) has had state-of-the-art evaluations in the last few decades in the areas of image classification, object detection, voice recognition, natural language processing, and so on. Reducing the number of parameters

in Artificial Neural Network (ANN) is the main advantage of CNN. The most important thing about CNN tasks is that they should not have spatially dependent features. Another vital CNN functionality is feature abstraction when input propagates to the deeper layers. i.e., in image recognition, the edge might be spotted in the first layers, and then the simpler figures in the second layers, and then the higher-level features such as faces in the third layers [52]. A standard CNN consists of one or more blocks of convolution and pooling layers, after those single or multiple fully connected layers and an output layer at the end as shown in Figure 3.3.

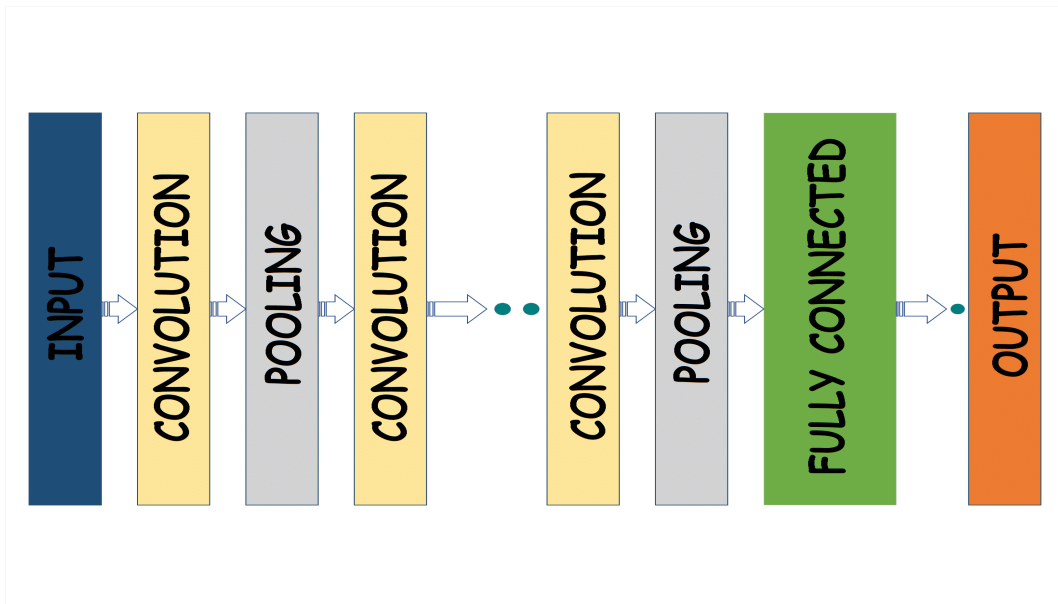


Figure 3.3: Workflow of a classical CNN.

3.5.1 Convolutional Layer

The central part of CNN is a convolutional layer. Images have a common fixed nature, which means the formation of one edge of the image is the same as any other edge. Therefore, a feature learned in one part can match the parallel shape in another. In a high dimensional image, we take a small region and proceed through all the edges in the high dimensional image (Input). During the process at edges, we convolute them into a single place (Output). Each small region that proceeds over the input or large image and learns different portions of an input image is known as the filter (Kernel). The kernels are later updated based on the backpropagation method [53]. The standard convolutional procedure has depicted in Figure 3.4.

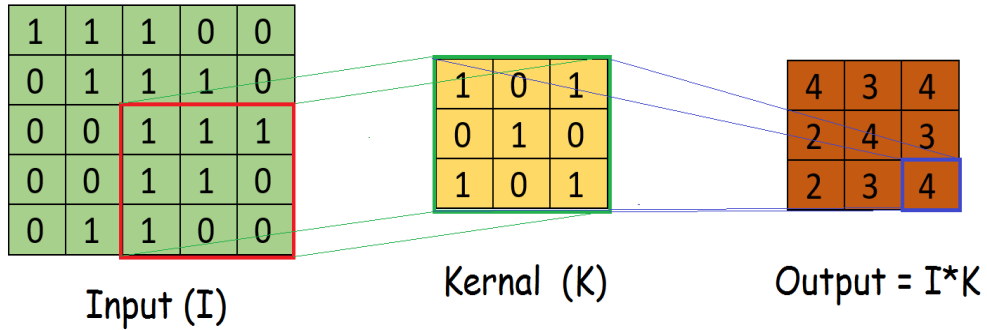


Figure 3.4: Workflow of a convolutional layer.

3.5.2 Pooling Layer

Pooling means downsampling of an image. It gets a small section of the convolutional layer's output as input and down-samples it to generate a solo output. There are various sub-sampling methods such as max pooling, average pooling, global average pooling, global max pooling, etc. Pooling minimizes the number of parameters to be computed [54]. Figure 3.5 shows operations of max and average pooling, which are the most famous pooling techniques.

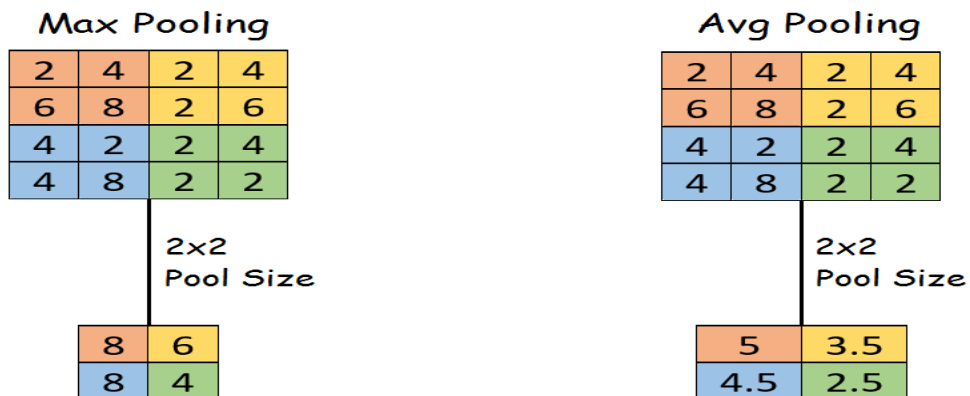


Figure 3.5: The pooling layer, also known as sub-sampling layer.

3.5.3 Fully Connected Layer

Fully connected layers are the last part of CNN as represented in Figure 3.6. This layer aims to take inputs from all neurons in the preceding layer and executes operation with each neuron in the current layer to generate output [55].

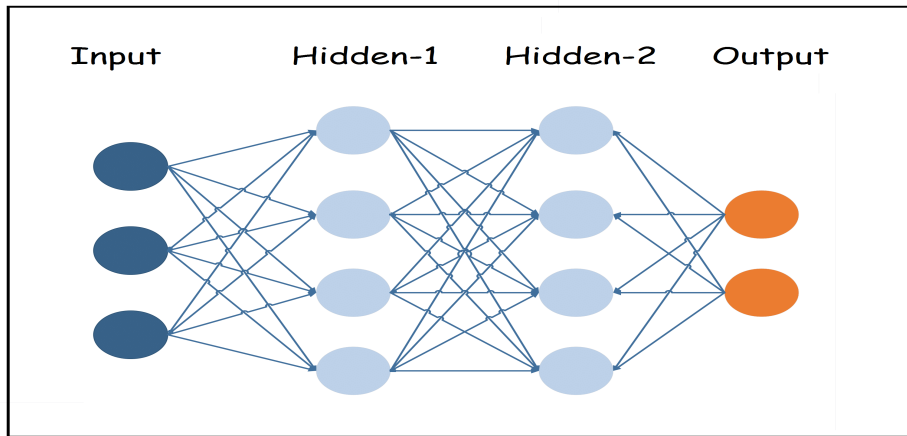


Figure 3.6: The fully-connected layer of a classical CNN.

3.6 Transfer Learning

Transfer learning using pre-trained CNN means applying an already trained model to a new task. It states that what learned in one configuration has been used to enhance optimization in another configuration [56]. Nowadays, it is very famous in deep learning because it can also train deep neural networks with small data. Transfer learning can be applied when there is a new dataset smaller than the dataset used to train the pre-trained architecture [57]. Many problems in the field of data science haven't millions of labeled data points available to train such a difficult model. Transfer learning is a useful method in which a model is trained and developed for one dataset and then re-used on a second related dataset. Without transfer learning, we train many models for each dataset separately. In the case of transfer learning, we use a pre-trained model again for a similar dataset [58]. Figure 3.7 demonstrates the key benefit of transfer learning.

Currently, three main techniques successfully employ CNNs to transfer learning [58, 56], such as:

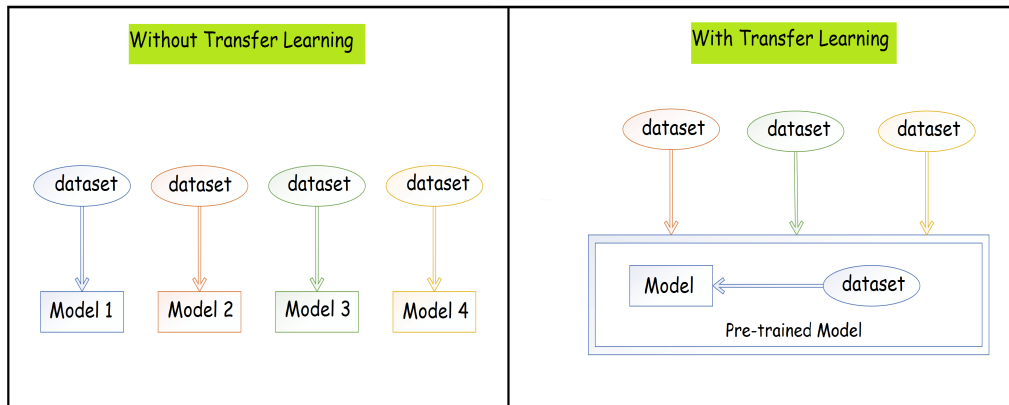


Figure 3.7: The concept of with and without transfer learning.

- Strategy 1:** Train the whole model from scratch. i.e., you only use the architecture of a pre-trained model and train it according to your dataset.
- Strategy 2:** Fine-tune the pre-trained convolutional networks to your dataset. Freeze the initial layers and continue backpropagation and update the parameters of layers.
- Strategy 3:** Freeze the convolutional base and use CNN as a feature extractor only. After that, throw away the classification part. Then, use additional machine learning algorithms such as SVM, KNN, etc., for the classification.

Table 3.4 shows the possibilities of when to use which strategy based on the dataset similarity and size [59]. Here, the similarity means how much our dataset is similar to the pre-trained model's dataset, and size indicates the volume of our dataset.

Table 3.4: When to use which strategy of pre-trained networks

Quadrant#	Dataset Size	Dataset Similarity	Strategy#
Quadrant 1	Large	Different	Strategy 1
Quadrant 2	Large	Similar	Strategy 2
Quadrant 3	Small	Different	Strategy 2
Quadrant 4	Small	Similar	Strategy 3

We are applying the second strategy of pre-trained networks to our dataset, which is most likely to our proposed problem.

3.7 CNN Architectures

There are many traditional and latest pretrained CNN architectures used for image processing tasks such as LeNet [60], AlexNet [61], VGG [62], Xception [63], ResNet [1], DenseNet [64], etc. In our proposed work, we apply five of CNN models for underwater image classification such as VGG16, VGG19, ResNet50, DenseNet121 and Xception.

3.7.1 VGGNet

Simonyan and Zisserman proposed a model by configuring AlexNet [61] deeper and named it as VGGNet. The researcher applied 3x3 filters in all the layers with the stride of 1 and made the network deeper keep other parameters unchanged. They have introduced six different CNN configurations such as VGG11, VGG11-LRN, VGG13, VGG16, VGG16 (Conv1x1), and VGG19 with 11, 11, 13, 16, 16, 19 weighted layers. The max-pooling applied in the pooling layer and 2x2 pooling size with a stride of 2. The ReLU function had used as an activation function in the activation layer and a SoftMax activation function in the dense layer. VGG16 network has approximately 138 million, and the VGG19 has 14.71 million parameters respectively [62]. We use VGG16 and VGG19 in our proposed model. The architecture of VGG16 and VGG19 as shown in Figure 3.8.

3.7.2 ResNet

Deep networks are hard to train because of the notorious vanishing gradient problem. However, this problem had tackled by normalized initialization. The deeper model demonstrates bad performances not because of overfitting but on both train and test errors. That specifies that loss reduction of the deeper model is too hard. The authors applied a pre-trained model with extra layers to execute identity mapping to control this issue. In such a manner, the performance of the pre-trained network and the deeper network should be the same. They have proposed a deep residual network to solve the degradation issue and named their model as ResNet [1]. The authors added residual mapping:

$$(H(x) = F(x) + x) \tag{3.1}$$

Instead of desired underlying mapping ($H(x)$) into their network as illustrated in Figure 3.9.

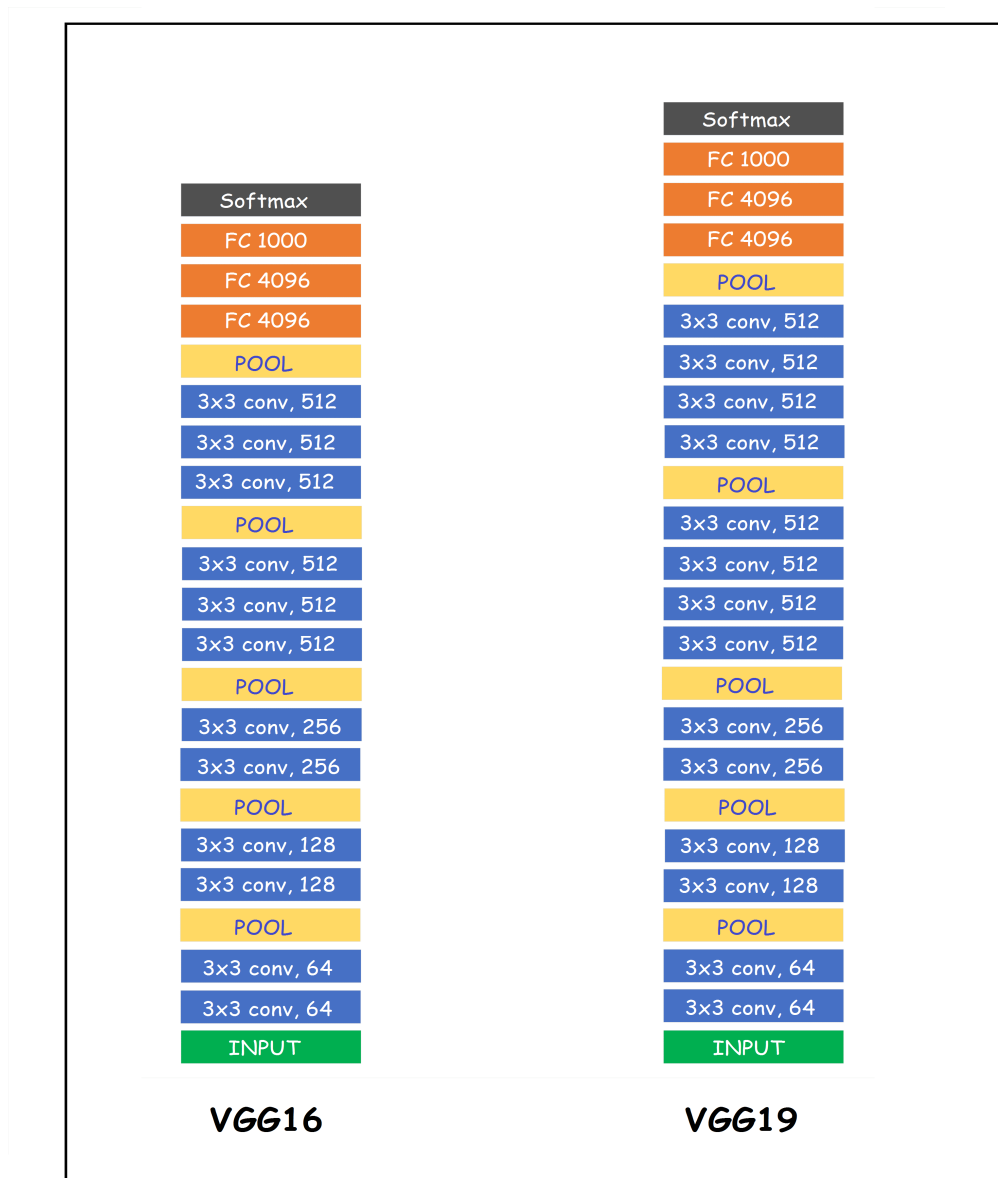


Figure 3.8: A visualization of VGGNet architecture.

ResNet is stacking the residual blocks together. Add skip connections to a plain network to convert it into a residual network. Each ResNet block is two layers deep (used in small residual networks such as ResNet18, 34) or three layers deep (used in large residual networks like ResNet 50,101,152).

3.7.3 DenseNet

Dense Convolution Networks (DenseNet) was presented by Huang et al. [64], which introduces Dense block in traditional CNN. A layer in a Dense block takes

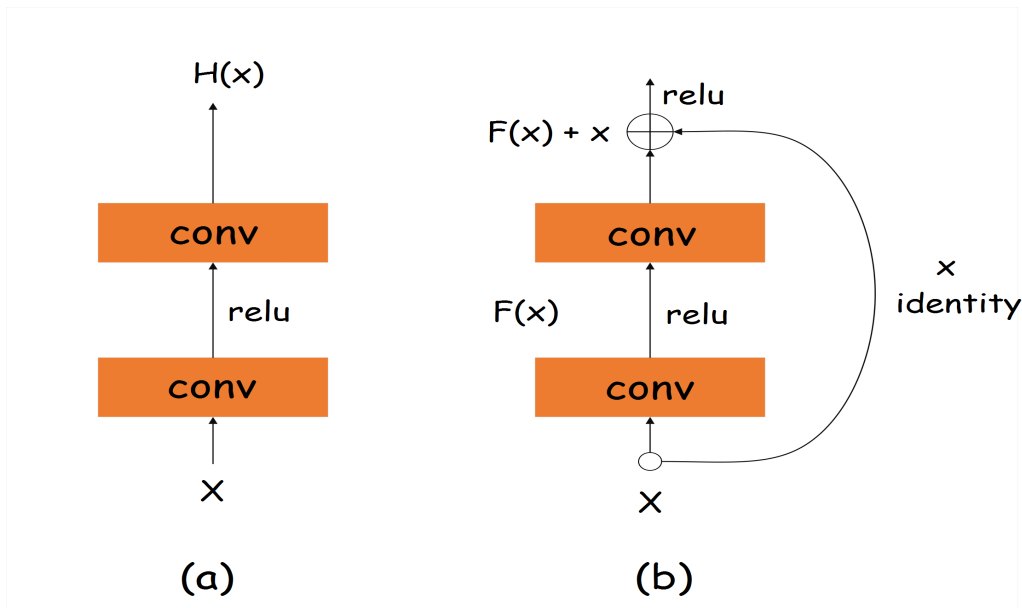


Figure 3.9: (a) Plain layer (b) Residual Block [1]

input from the concatenation of the output of all the previous layers. Each layer in a dense block reuses the results of all preceding layers, minimizing vanishing gradient problems and stabilizing feature propagation. DenseNet layers use a small number of filters, and they add a small set of new feature maps. DenseNets are similar to ResNets, but instead of sum, DenseNet concatenates the output feature maps of the layer with the incoming feature maps.

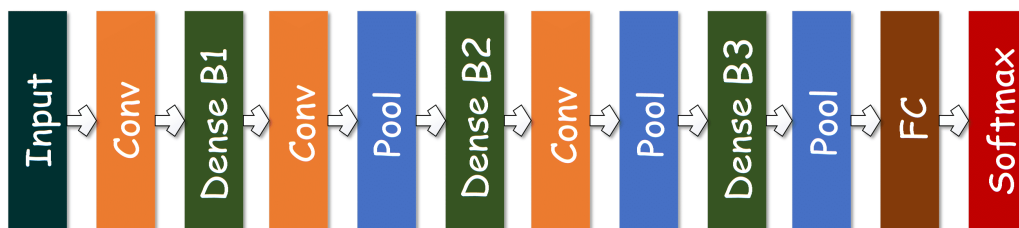


Figure 3.10: The standard structure of a Dense block with three layers.

Figure 3.10 demonstrates a Dense network with three dense blocks. The non-linear conversion functions in a Dense block are a composite function of ReLU, batch normalization, and 3×3 convolution operation. Also, the 1×1 bottleneck layer is used in DenseNet to reduce the spatial dimensions.

3.7.4 Xception

Xception is the extreme version of Inception [65]. It is even better than Inception-v3 [66], with an updated depth-wise separable convolution. A classic convolutional network in which convolutional layers examine correlations across both space and dept. In Xception, the input image has split into many compressed parts. It maps the spatial correlation for each output separately. Then executes 1x1 depth-wise convolution to find a cross-channel correlation. The depth-wise separable convolution has followed by point-wise convolution. Depth-wise convolution is the channel-wise ($n \times n$) spatial convolution. The amount of ($n \times n$) spatial convolution depends on the number of channels, i.e., five channels = 5 ($n \times n$) spatial convolutions. Point-wise convolution is the 1×1 convolution to change the dimension [63]. Figure 3.11 represents the workflow of Xception.

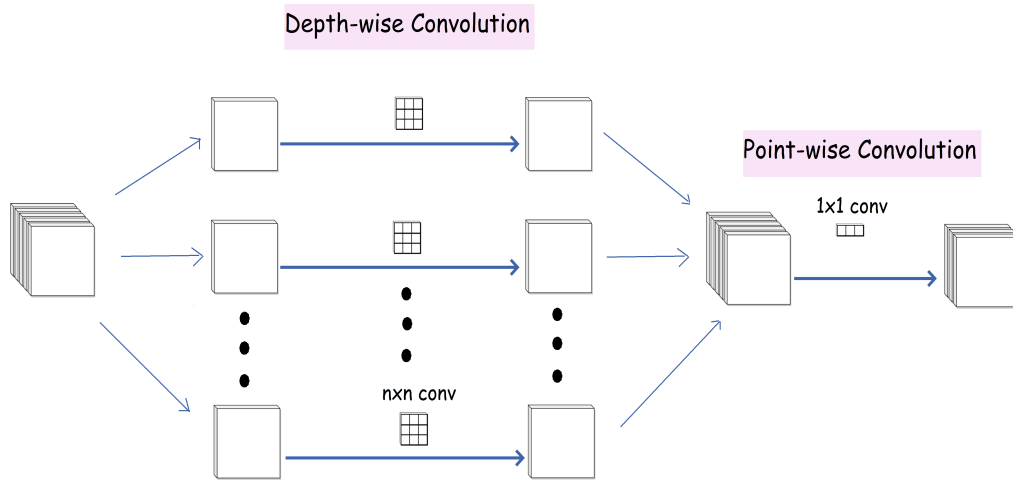


Figure 3.11: The structure of Xception architecture and dept-wise separable convolutions.

3.7.5 CNN Architectures: an overview

The pre-trained networks trained on more than a million images from the ImageNet database contest into the 1000 different classes [61]. Configuration of various CNN architectures shown in Table 3.5 with advantages and disadvantages of each network.

Table 3.5: A typical architectures in convolutional neural networks

Model	Year	Layer#	Param#	Merits and Demerits
VGG16/VGG19	2014	16/19	138/143m	It learns more complex features at a lower cost using multiple non-linear layers but is slow to train, and the pre-trained weights are sizeable in terms of disk and bandwidth [67].
ResNet50	2015	50	25.6m	It tackles the vanishing gradient issue using identity mapping, and the skip level connections make dimensionality complex between different layers [68].
DenseNet121	2016	121	8m	Reinforce feature propagation, lessen the number of parameters, reduce the vanishing-gradient problem, etc. The extreme contacts reduce computation-efficient and make more prone to overfitting [69].
Xception71	2017	71	23m	More efficient regarding computation time [70].

3.8 Evaluation Measurement

When we are training a model in deep learning, we have to choose a correct classification problem, label encoding, activation and loss functions, and accuracy metrics [71].

3.8.1 Classification Problem and Label Encoding

First, we have to understand the classification problem we are solving. There are three main classification types in machine learning, such as binary classification, multi-class classification, and multi-label classification.

Binary Classification

A binary classification is for one or two target classes. For example, (i) Is it Man in the image? (ii) Is it Man or Woman in the Image? A floating number of 0.0 for Man and 1.0 for Woman label in the case of binary classification.

Multi-class Classification

When we have more than two classes, only one target assign to an input, i.e., Which car is in the image: BMW, Mercedes, Range Rover, Audi? For this problem, one-hot encoding can be an appropriable solution as shown below:

$$\text{BMW} = [1 \ 0 \ 0 \ 0]$$

$$\text{Mercedes} = [0 \ 1 \ 0 \ 0]$$

$$\text{Range Rover} = [0 \ 0 \ 1 \ 0]$$

$$\text{Audi} = [0 \ 0 \ 0 \ 1]$$

Also, we can use a vector array of integers for multi-class classification problem:

$$\text{BMW} = [1]$$

$$\text{Mercedes} = [2]$$

$$\text{Range Rover} = [3]$$

$$\text{Audi} = [4]$$

Multi-label Classification

If we have more than two classes, multiple classes assign to an input. For example, Which car is in the image: BMW, Mercedes, Range Rover, Audi? Multi-hot encoding will be used for these kinds of tasks, as shown below:

$$\text{BMW} = [1 \ 0 \ 0 \ 0]$$

$$\text{BMW,Mercedes} = [1 \ 1 \ 0 \ 0]$$

$$\text{Range Rover} = [0 \ 0 \ 1 \ 0]$$

$$\text{Audi,Mercedes} = [0 \ 1 \ 0 \ 1]$$

In our case, the Fishes in the Wild dataset contains many images that are either single or multiple fishes. Therefore, we have labeled these images using multi-hot encoding or multi-label classification techniques.

3.8.2 Activation Function

The neural network has neurons that operate using weight, bias, and their respective activation function. The weights and biases of the neurons would update based on the error rate at the output. This task is known as backpropagation. Activation functions have a key role in backpropagation since the gradients are provided together with the error to update biases and weights [72]. There are several layer activation functions available in deep learning. We are discussing three of them as Sigmoid [73], Softmax [74] and ReLU [75, 76]. ReLU applies as an activation function in the hidden layers of pre-trained CNN architectures. Our proposed work is a multilabel binary classification problem. Therefore we implement the sigmoid and

softmax functions in the output layers.

Sigmoid Activation Function

In this activation function, the result is either 0 or 1 and commonly used in the output layer of binary classification tasks. Results can be easily predicted using a threshold i.e., if the value is less than 0.5 then it would be 0 otherwise 1. The formula of sigmoid activation function is given below:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \quad (3.2)$$

Softmax Activation Function

Softmax is similar to the sigmoid function but is useful when we are trying to do multiclass or multilabel classification problems. The output of the input vectors would lie between 0 and 1, and would also divide by the total number of outputs. Softmax function formula is given below:

$$Softmax(x)_i = \frac{exp(x_i)}{\sum_j exp(x_j)} \quad (3.3)$$

ReLU Activation Function

ReLU has commonly been used as an activation function in the hidden layers of neural networks. It gives an output of 1 if the value is positive otherwise is 0. ReLU is less computationally expensive and learns much faster than other activation functions because it contains simpler mathematical operations. The formula of ReLU has given below:

$$ReLU = \max(0, X) \quad (3.4)$$

3.8.3 Loss Function

Loss functions are an approach to measuring how far a predicted value is from the actual value. There are no suitable loss functions to algorithms in machine learning that can use in all circumstances. Many loss functions are available in regression and classification tasks. We are discussing the ones used in our proposed classification task.

Binary Crossentropy

Binary cross-entropy is a loss function that computes the cross-entropy loss between actual and the predicted output. Binary cross-entropy can use in two conditions: (i) If there are only two classes labeled as 0 and 1. There should be

only a floating-point value per prediction for each instance. (ii) If there are equal to or more than two classes with multi-hot encoded labels. There should be only a floating-point value per label for each instance [77].

Categorical Crossentropy

It also computes the cross-entropy loss between actual outputs and the predicted outputs. Categorical cross-entropy can be used in a condition if there are two or more labeled classes [78].

3.8.4 Confusion Matrix

A confusion matrix has considered one of the best evaluation tools for classification tasks. It can use in binary classification as well as for multiclass classification tasks. A confusion matrix represents the performance of a classification model. The standard structure of the confusion matrix has shown in Figure 3.12.

The diagram shows a 2x2 confusion matrix. The columns are labeled 'Actual Class' (Positive, Negative) and the rows are labeled 'Predicted Class' (Positive, Negative). The cells contain TP, FP, FN, and TN. The TP and TN cells are green, while the FP and FN cells are orange.

		Actual Class	
		Positive	Negative
Predicted Class	Positive	TP	FP
	Negative	FN	TN

Figure 3.12: Confusion matrix or the table of confusion.

The confusion matrix has four main attributes used to state the measurement metric of the model, such as:

TP (True Positive): Number of correct predictions when the actual class was positive.

TN (True Negative): Number of correct predictions when the actual class was negative.

FP (False Positive): Number of incorrect predictions when the actual class was positive.

FN (False Negative): Number of incorrect predictions when the actual class was negative.

The accuracy, precision, recall, and F1 score are characteristics of an evaluation metrics of a model. These characteristics are measured based on the TP, TN, FP, and FN [79, 80].

3.8.5 Accuracy Metrics

In the proposed method, we measure our models based on accuracy metrics.

Accuracy

The accuracy metric is a calculation of how near the predicted value is to the actual value of the quantity. If an error is low, the evaluation is correct. The accuracy metric evaluates the rate of accurate predictions over the total number of samples measured [81]. The formula for accuracy metric is:

$$Acc = \frac{(TP + TN)}{(TP + TN) + (FP + FN)} \quad (3.5)$$

Binary Accuracy

Binary accuracy evaluates how often predictions be similar to binary labels. The binary accuracy can be used as a measurement metric in two types of tasks as binary classification and multi-label classification. The metric encoded actual labels in one-hot or multi-hot vectors. Also, it creates two local variables that measure the rate with which the predicted label matches the actual label. This rate is known as binary accuracy, in which a static task divides the total variable by the count variable [82].

Categorical Accuracy

The Categorical accuracy evaluates how often predictions be similar to one-hot labels. The metric has commonly used in a multi-class classification problem as an evaluation metric. Also, it creates two local variables that measure the rate with which the predicted label matches the actual label. This rate is known as Categorical accuracy, in which a static operation divides the total variable by the count variable [82].

CHAPTER 4

DATA ANALYSIS, RESULTS & FINDINGS

This chapter, gives the evaluation of this work. We discuss the dataset used for this study followed by experimental settings and finally we discuss the experimental results for the proposed approach.

4.1 Experimental Setup and Dataset

All networks were implemented using the TensorFlow framework and trained by Tesla P100-PCIe GPUs. The experiments had done in both desktop-based and cloud-based environments. The tools and technologies used for the groundwork of the proposed work have displayed in Table 4.1. These tools have a significant value while implementing image classification problems.

In this work, we applied classification algorithms for fish species recognition. We labeled the images into three classes as Rockfish, Starfish, and Tilefish. The average image size is 0.14mp and, 415x361 is the median image ratio. From the selected 767 images, 289 are from the Tilefish class, which is approximately 38%. There are 181 images of Starfish and 170 Rockfish images, which are 23% and 22% respectively. The remaining 17% of 127 images are multi-labeled such as 118 images of Rochfish+Starfish, and nine images of Rockfish+TileFish. Figure 4.1 presents the percentage and comparative analysis of the contribution based on the number of images.

For this purpose, we have selected 767 images and manually annotated them by applying a multi-label classification technique using an online tool [83]. A total of 894 annotations have been done for above mentioned three classes. The class balance of the annotated labels have shown in Table 4.2.

Further, the dataset was converted into two versions of 767 images, i.e., (i) Raw or the original form, and (ii) Enhanced form. Table 4.3 presents various data preprocessing and augmentation techniques on behalf of raw and enhanced versions.

Table 4.1: Most used tools and technologies in the proposed research

Tool	Technology	Environment	Objective
Keras	Python	Both	A Python API that offers an interface for ANNs.
Tensorflow	Python	Both	It is an end-to-end python library for machine learning and artificial intelligence.
Numpy	Python	Both	It is a python library used for working with arrays and provides comprehensive mathematical functions.
Pandas	Python	Both	It is a python library used for data manipulation and analysis.
Jupyter	Anaconda	Desktop	A web-based interactive computing platform.
VS Code	Microsoft	Desktop	A code editor for debugging advanced web and cloud applications.
Origin	OriginLab	Desktop	Used for data visualization.
MS Visio	Microsoft	Desktop	Used for making and designing graphs.
Colab	Google	Cloud	Allows us to write and execute arbitrary python code through the browser.
P100 GPU	Tesla	Cloud	Can process many pieces of data at once and make them useful for classification.

Table 4.2: The class balance of the dataset

Class	Annotations
Rockfish	297
Starfish	299
Tilefish	298

The enhanced version is done Contrast Limited Adaptive Histogram Equalization (CLAHE) [38, 50, 51]. It's an image contrast method for low-resolution photos.

We applied CLAHE on Fishes in the Wild dataset. A sample image has shown in Figure 4.2.

The Fishes in the wild dataset consists of 767 images labeled with 894 annotations for Raw and Enhanced versions each. We have done 121 experiments using five

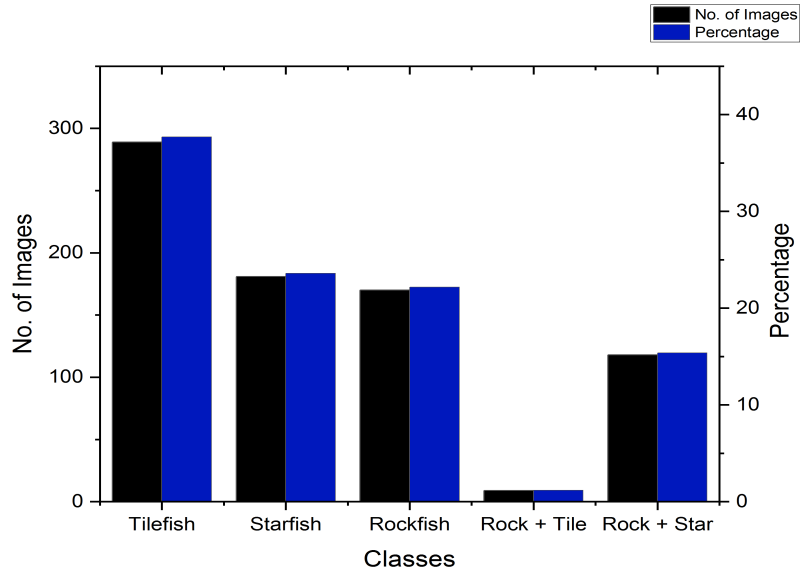


Figure 4.1: An analysis of fish species.

Table 4.3: Data Preprocessing

	Raw	Enhanced
Auto-orient	Applied	Applied
Auto-adjust contrast	Not applied	Using adaptive equalization
Augmentations	Not applied	Not applied
No. of Images	767	767
Annotations	894	894

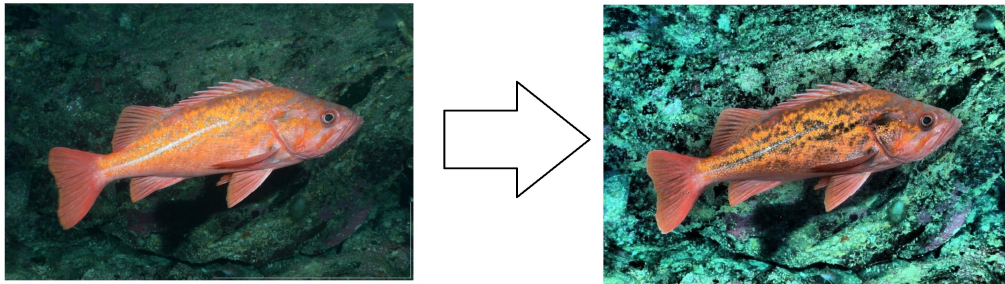


Figure 4.2: Before and after applying the CLAHE method.

CNN architectures on the datasets. For this task, we have applied VGG16, VGG19, ResNet50, DenseNet121, and Xception71. Some hyper-parameters have not changed since they remain constant in every experiment. Constant parameters have shown

in Table 4.4.

Table 4.4: The constant parameters

Parameter	Value
dept	3
weights (Pretrained)	imagenet
include_top	false
trainable	false
optimizer	adam

Dept refers to the number of channels in an input image. In our case, we are using colored images with three channels as Red, Green, and Blue (the dept would be 1 in case of a Grayscale image). Similarly, we are using pre-trained CNN models that trained on ImageNet[10] database. Include top is a hyperparameter that has included the ImageNet classifier at the top. But we created our model and changed it to false for all experiments to not include the ImageNet classifier. Trainable is a Boolean hyperparameter. Configuring “trainable=false” converts all the layer’s weights to non-trainable. This process is known as “freezing” the layer.

4.2 VGG16 as a Model

We have done 16 experiments on the original or raw version as shown in Table 4.5. Experiment 12 gives more precise results than other experiments. Firstly, we split data into 70% for training and 30% for testing. Then, we assigned 320x320x3 to the initial input shape and then set the sigmoid activation function at the output layer. This experiment has trained in one hour and 17 minutes, and 36 seconds, and tested in 788 milliseconds in which the number of train batches was nine and test batches were four per iterations. In the end, it gives 95.38% of accuracy and 0.1411 of loss.

Similarly, 13 experiments have been applied on the raw version using VGG16. Details for each experiment have listed in Table 4.6. Experiment 10 provides higher accuracy than other implementations. Dataset has split into 80% and 20% for train and test. We feed 128x128x3 as input and use the sigmoid activation function at the output layer. We manually set the batch size to 64. Experiment 13 takes 10 minutes to train and 3 seconds to test. The Auto-tuned method provides ten batches for train and three batches for the test. It offers 96.10% accuracy and 0.1197 loss and correctly predicted 8 out of 10 images in 3 seconds. Also, we applied the Categorical Crossentropy loss function instead of the Binary Crossentropy loss function in experiment 8. But, it increases an error rate as compared to other

Table 4.5: VGG16 on raw version

Exp	Input	Split		Batch			Activation	Epochs	Time			Loss	Accuracy
		Train	Test	Size	Train	Test			Train	Test	Predict		
1	64x64x3	80%	20%	64	10	3	Sigmoid	100	2m 37s	42ms	5s	0.2164	0.9199
2	64x64x3	80%	20%	64	10	3	Sigmoid	300	3m 22s	45ms	5s	0.1782	0.9134
3	256x256x3	80%	20%	128	5	2	Sigmoid	100	5m 34s	144ms	6s	0.2810	0.9069
4	128x128x3	80%	20%	64	10	3	Sigmoid	200	3m 20s	96ms	4s	0.1868	0.9069
5	320x320x3	80%	20%	64	10	3	Sigmoid	200	11m 4s	483ms	3s	0.2071	0.9242
6	320x320x3	70%	30%	64	9	4	Sigmoid	500	28m 3s	784ms	7s	0.1609	0.9380
7	128x128x3	80%	20%	64	10	3	Softmax	200	4m 11s	99ms	5s	0.2237	0.8961
8	128x128x3	80%	20%	64	10	3	Softmax	200	5m 4s	102ms	5s	0.5015	0.9004
9	128x128x3	80%	20%	64	10	3	Sigmoid	500	6m 44s	99ms	5s	0.1776	0.9177
10	128x128x3	80%	20%	64	10	3	Sigmoid	1000	11m 8s	102ms	5s	0.2002	0.9177
11	320x320x3	70%	30%	64	9	4	Sigmoid	1000	52m 10s	796ms	8s	0.1420	0.9524
12	320x320x3	70%	30%	64	9	4	Sigmoid	1500	1h 17m 36s	788ms	4s	0.1411	0.9538
13	320x320x3	70%	30%	64	9	4	Sigmoid	100	21m 38s	3ms	1s	0.3033	0.8701
14	64x64x3	80%	20%	64	10	3	Sigmoid	500	4m 7s	42ms	5s	0.1812	0.9091
15	32x32x3	85%	15%	64	11	2	Sigmoid	500	4m 40s	26ms	5s	0.2424	0.9080
16	352x352x3	80%	20%	64	10	3	Sigmoid	100	8m 12s	588ms	8s	0.2651	0.9380

implementations, as shown in Table 4.6. It causes too many wrong predictions, such as correctly predicting 4 out of 10 images.

Table 4.6: VGG16 on enhanced version

Exp	Input	Split		Batch			Activation	Epochs	Time			Loss	Accuracy
		Train	Test	Size	Train	Test			Train	Test	Predict		
1	64x64x3	80%	20%	64	10	3	Sigmoid	100	5m 7s	93ms	4s	0.2170	0.9221
2	64x64x3	80%	20%	64	10	3	Sigmoid	300	7m 52s	96ms	4s	0.1886	0.9329
3	256x256x3	80%	20%	128	5	2	Sigmoid	100	4m 56s	93ms	3s	0.2513	0.9134
4	128x128x3	80%	20%	64	10	3	Sigmoid	200	3m 20s	96ms	4s	0.1622	0.9394
5	320x320x3	70%	30%	64	9	4	Sigmoid	200	11m 23s	1s	4s	0.1978	0.9221
6	320x320x3	70%	30%	64	9	4	Sigmoid	500	26m 50s	1s	3s	0.1507	0.9408
7	128x128x3	80%	20%	64	10	3	Softmax	200	4m 0s	93ms	3s	0.1871	0.9177
8	128x128x3	80%	20%	64	10	3	Softmax	200	3m 20s	96ms	3s	0.4427	0.9177
9	128x128x3	80%	20%	64	10	3	Sigmoid	500	5m 49s	99ms	3s	0.1263	0.9567
10	128x128x3	80%	20%	64	10	3	Sigmoid	1000	10m 3s	96ms	3s	0.1197	0.9610
11	128x128x3	80%	20%	64	10	3	Sigmoid	6000	52m 33s	96ms	3s	0.2781	0.9481
12	128x128x3	80%	20%	64	10	3	Sigmoid	2000	19m 32s	99ms	2s	0.1440	0.9567
13	128x128x3	80%	20%	64	10	3	Sigmoid	1500	14m 41s	99ms	4s	0.1291	0.9567

4.3 VGG19 as a Model

Fourteen tests have been implemented on the raw version using the VGG19 network. Experiment 13 provides 95.02% accuracy with a loss of 0.1777. We divide the dataset to 80% for training and 20% for testing. We set the other parameters like input_shape to 256x256x3, batch to 64, train batches to 10 and test batches to 3, and fixed sigmoid at the output layer. It takes 36 minutes and 16 seconds to train in 1000 iterations and 378ms to test using Tesla P100 GPU on Google Colab. Correctly predicts 6 out of 10 images because of over-fitting. We fixed the number of epochs to 100 and tried again. It has given the same accuracy as experiment 13 with better loss and prediction. It predicted 8 out of 10 correctly. In experiments 4 and 5, we fix softmax activation at the output layer, but it gives 90% and 91%

accuracies. The offered accuracies are too low as compared to experiment 13. Table 4.7 shows all the details of each experiment that has been done on the raw version dataset using VGG19.

Table 4.7: VGG19 on raw version

Exp	Input	Split		Batch			Activation	Epochs	Time			Loss	Accuracy
		Train	Test	Size	Train	Test			Train	Test	Predict		
1	224x224x3	80%	20%	64	10	3	Sigmoid	100	4m 39s	308ms	4s	0.2433	0.9026
2	224x224x3	80%	20%	64	10	3	Sigmoid	500	15m 50s	306ms	3s	0.1798	0.9351
3	224x224x3	80%	20%	64	10	3	Sigmoid	1000	30m 21s	306ms	4s	0.1739	0.9372
4	128x128x3	80%	20%	64	10	3	Softmax	100	2m 43s	114ms	3s	0.2370	0.9113
5	128x128x3	80%	20%	64	10	3	Softmax	500	6m 37s	120ms	3s	0.2232	0.9048
6	128x128x3	80%	20%	64	10	3	Sigmoid	100	2m 38s	111ms	3s	0.2318	0.9091
7	128x128x3	80%	20%	64	10	3	Sigmoid	500	6m 35s	117ms	3s	0.1896	0.9221
8	128x128x3	80%	20%	64	10	3	Sigmoid	1000	12m 15s	120ms	3s	0.1963	0.9199
9	64x64x3	80%	20%	64	10	3	Sigmoid	100	2m 5s	48ms	3s	0.2353	0.8896
10	320x320x3	80%	20%	64	10	3	Sigmoid	100	7m 16s	579ms	5s	0.2607	0.9004
11	256x256x3	80%	20%	64	10	3	Sigmoid	100	5m 42s	375ms	7s	0.2598	0.9048
12	256x256x3	80%	20%	64	10	3	Sigmoid	500	19m 31s	381ms	3s	0.1854	0.9416
13	256x256x3	80%	20%	64	10	3	Sigmoid	1000	36m 16s	378ms	3s	0.1777	0.9502
14	256x256x3	80%	20%	64	10	3	Sigmoid	10000	5h 49m 47s	375ms	5s	0.4404	0.9307

We did nine experiments on the enhanced version of the dataset. Experiment 7 gives more accurate results than other assessments. All the hyperparameters are the same as experiment 13 of the raw version. But, the only difference is the number of epochs, such as 500 epochs in the enhanced version. The evaluation takes 18m 46s to train the model. In the end, it gives 0.1163 test loss and 96.10% test accuracy. Table 4.8 shows all experiments records.

Table 4.8: VGG19 on enhanced version

Exp	Input	Split		Batch			Activation	Epochs	Time			Loss	Accuracy
		Train	Test	Size	Train	Test			Train	Test	Predict		
1	224x224x3	80%	20%	64	10	3	Sigmoid	100	4m 30s	315ms	3s	0.1959	0.9394
2	224x224x3	80%	20%	64	10	3	Sigmoid	500	15m 50s	318ms	3s	0.1222	0.9567
3	224x224x3	80%	20%	64	10	3	Sigmoid	1000	30m 50s	306ms	3s	0.1194	0.9437
4	256x256x3	60%	40%	64	8	5	Sigmoid	100	6m 28s	805ms	1s	0.2157	0.9251
5	256x256x3	60%	40%	64	8	5	Sigmoid	500	19m 9s	800ms	4s	0.1411	0.9511
6	256x256x3	80%	20%	64	10	3	Sigmoid	100	5m 41s	375ms	3s	0.1921	0.9394
7	256x256x3	80%	20%	64	10	3	Sigmoid	500	18m 46s	375ms	3s	0.1163	0.9610
8	256x256x3	80%	20%	64	10	3	Sigmoid	800	29m 3s	372ms	3s	0.1105	0.9567
9	256x256x3	80%	20%	64	10	3	Sigmoid	1000	35m 56s	372ms	3s	0.1108	0.9524

4.4 ResNet50 as a Model

ResNet50 gives poor results on behalf of comparison with other CNN architectures. The best experiment for the raw version is experiment 7, with 87.88% accuracy. One thing we noted is that increasing the input shape reduces the accuracy. So, we feed 100x100x3 as input, set the batch size to 64, and sigmoid activation function at the output layer. The model has trained in 9 minutes and 16 seconds

Table 4.9: ResNet50 on raw version

Exp	Input	Split		Batch			Activation	Epochs	Time			Loss	Accuracy
		Train	Test	Size	Train	Test			Train	Test	Predict		
1	512x512x3	70%	30%	64	9	4	Sigmoid	100	11m 48s	1s 684ms	4s	0.3981	0.8240
2	512x512x3	80%	20%	64	10	3	Sigmoid	100	11m 21s	4s	4s	0.4003	0.8116
3	320x320x3	80%	20%	64	10	3	Sigmoid	100	5m 41s	3s 435ms	4s	0.4175	0.8052
4	320x320x3	80%	20%	64	10	3	Sigmoid	500	21m 48s	3s 435ms	4s	0.3359	0.8571
5	320x320x3	80%	20%	64	10	3	Sigmoid	1000	42m 11s	435ms	4s	0.3118	0.8571
6	448x448x3	80%	20%	64	10	3	Sigmoid	1000	1h 18m 47s	798ms	4s	0.3044	0.8636
7	100x100x3	80%	20%	64	10	3	Sigmoid	1000	9m 16s	93ms	4s	0.3149	0.8788
8	256x256x3	80%	20%	64	10	3	Sigmoid	1000	29m 27s	385ms	4s	0.3179	0.8485

after 1000 iterations. The model correctly predicted 7 out of 10 species in 3 seconds. Table 4.9 presents all eight evaluations applied on a raw version using ResNet50.

We performed 16 experiments on the enhanced version dataset using the ResNet50 model. In experiment 4, we implemented Flatten() class in the output layer instead of GlobalAveragePooling2D() and assessed the poorest result of our thesis as shown in Table 4.10. The accuracy we attained is 55.19% and a loss of 6.2860. The model has predicted 3 out of 10 accurately. However, the best experiment is 16, which gives 93.29% of accuracy. The model has trained in 53 minutes and 6 seconds, and it predicted 7 out of 10 species in 5 seconds. Table 4.10 shows all the results comprehensively.

Table 4.10: ResNet50 on enhanced version

Exp	Input	Split		Batch			Activation	Epochs	Time			Loss	Accuracy
		Train	Test	Size	Train	Test			Train	Test	Predict		
1	256x256x3	80%	20%	64	10	3	Sigmoid	100	4m 30s	291ms	4s	0.3190	0.8766
2	256x256x3	80%	20%	64	10	3	Sigmoid	500	15m 35s	297ms	3s	0.2444	0.9113
3	256x256x3	80%	20%	64	10	3	Sigmoid	1000	29m 47s	300ms	4s	0.2225	0.9177
4	256x256x3	80%	20%	64	10	3	Sigmoid	100	4m 23s	294ms	5s	6.2860	0.5519
5	256x256x3	80%	20%	64	10	3	Softmax	100	5m 3s	294ms	4s	0.3267	0.8550
6	256x256x3	80%	20%	64	10	3	Softmax	500	15m 36s	297ms	4s	0.2494	0.8896
7	320x320x3	80%	20%	64	10	3	Sigmoid	500	22m 27s	335ms	4s	0.2184	0.9113
8	128x128x3	80%	20%	64	10	3	Sigmoid	500	6m 59s	108ms	4s	0.3416	0.8528
9	128x128x3	75%	25%	64	9	3	Sigmoid	1000	11m 31s	141ms	4s	0.3308	0.8594
10	416x416x3	80%	20%	64	10	3	Sigmoid	100	8m 23s	711ms	4s	0.2563	0.9026
11	512x512x3	80%	20%	64	10	3	Sigmoid	100	11m 32s	1s	4s	0.2543	0.9091
12	608x608x3	80%	20%	64	10	3	Sigmoid	100	15m 29s 1s	464ms	3s	0.2451	0.9113
13	800x800x3	80%	20%	64	10	3	Sigmoid	100	29m 59s 2s	523ms	6s	0.2636	0.9134
14	992x992x3	80%	20%	64	10	3	Sigmoid	100	42m 53s	3s	5s	0.2829	0.8983
15	608x608x3	80%	20%	64	10	3	Sigmoid	500	1h 12m 42s	1s 470ms	5s	0.2027	0.9286
16	512x512x3	80%	20%	64	10	3	Sigmoid	500	53m 6s	1s	5s	0.2080	0.9329

4.5 Xception71 as a Model

In this section, we are discussing the second most precise results. Table 4.11 shows that 11 experiments have been done on the raw version using Xception71. In experiment 4 and 5, we can see that small input shape such as 128x128x3 has given poor performances regarding accuracy. Experiments 9, 10, and 11 offered the same accuracy of 97.11%. However, 10 and 11 are over-fitted models, as we can see

the error rates. So, experiment 9 is considered the more accurate assessment. The hyperparameters had configured in such a manner that assigned 256x256x3 to input shape, split data into 70%/30% for train/test, set the batch size to 64, and fixed sigmoid function at the output layer. It trained in 5 min and 52 seconds. Here, the prediction rate is 90% (9 out of 10 correctly).

Table 4.11: Xception71 on raw version

Exp	Input	Split		Batch			Activation	Epochs	Time			Loss	Accuracy
		Train	Test	Size	Train	Test			Train	Test	Predict		
1	256x256x3	80%	20%	64	10	3	Sigmoid	100	5m 50s	387ms	7s	0.1518	0.9610
2	256x256x3	80%	20%	64	10	3	Sigmoid	500	20m 37s	381ms	4s	0.2273	0.9654
3	416x416x3	80%	20%	64	10	3	Sigmoid	100	11m 35s	993ms	7s	0.1207	0.9610
4	128x128x3	80%	20%	64	10	3	Sigmoid	100	2m 36s	126ms	4s	0.1441	0.8528
5	128x128x3	80%	20%	64	10	3	Sigmoid	500	7m 2s	126ms	4s	0.2215	0.9437
6	288x288x3	80%	20%	64	10	3	Sigmoid	100	6m 15s	486ms	2s	0.1521	0.9567
7	288x288x3	80%	20%	64	10	3	Sigmoid	500	26m 19s	486ms	4s	0.2194	0.9610
8	224x224x3	80%	20%	64	10	3	Sigmoid	100	4m 35s	303ms	2s	0.1389	0.9589
9	256x256x3	70%	30%	64	9	4	Sigmoid	100	5m 52s	616ms	4s	0.1315	0.9711
10	256x256x3	70%	30%	64	9	4	Sigmoid	500	19m 27s	612ms	6s	0.1917	0.9711
11	256x256x3	70%	30%	64	9	4	Sigmoid	1000	37m 34s	608ms	3s	0.2426	0.9711

Experiment 1 and 2 shows the same results of 98.70% accuracy. If we look at training duration, experiment 1 takes 5 minutes 12 seconds on 100 epochs to train. And experiment 2 takes 20 minutes and 54 seconds on 500 epochs. Similarly, experiment 1 has a 0.0621, and experiment 2 has a 0.0700 testing error rate. So, analyzing these experiment 1 has been considered the best model in Xception71. The other details of all experiments are in Table 4.12.

Table 4.12: Xception71 on enhanced version

Exp	Input	Split		Batch			Activation	Epochs	Time			Loss	Accuracy
		Train	Test	Size	Train	Test			Train	Test	Predict		
1	256x256x3	80%	20%	64	10	3	Sigmoid	100	5m 12s	384ms	4s	0.0621	0.9870
2	256x256x3	80%	20%	64	10	3	Sigmoid	500	20m 54s	375ms	4s	0.0700	0.9870
3	416x416x3	80%	20%	64	10	3	Sigmoid	100	10m 58s	984ms	4s	0.0662	0.9762
4	128x128x3	80%	20%	64	10	3	Sigmoid	100	2m 44s	123ms	4s	0.1096	0.9632
5	128x128x3	80%	20%	64	10	3	Sigmoid	500	6m 57s	123ms	4s	0.1764	0.9654
6	288x288x3	80%	20%	64	10	3	Sigmoid	100	6m 42s	484ms	4s	0.0646	0.9740
7	288x288x3	80%	20%	64	10	3	Sigmoid	500	26m 59s	489ms	3s	0.0752	0.9740
8	224x224x3	80%	20%	64	10	3	Sigmoid	100	4m 35s	306ms	2s	0.1089	0.9589
9	256x256x3	70%	30%	64	9	4	Sigmoid	100	5m 53s	616ms	2s	0.0763	0.9683
10	256x256x3	70%	30%	64	9	4	Sigmoid	500	19m 27s	616ms	4s	0.0784	0.9726
11	256x256x3	70%	30%	64	9	4	Sigmoid	1000	37m 29s	612ms	3s	0.0922	0.9740

4.6 DenseNet121 as a Model

DenseNet121 offers the most accurate predictions for raw and enhanced versions. Table 4.13 shows 12 implementations based on raw dataset using DenseNet121. The last experiment provides best results than other executions of DenseNet121 also other CNN architectures. Firstly, the dataset has divided into 80% and 20% for train

and test. Then, fixed batch size to 64 and set sigmoid function at the output layer. After that, we feed 448x448x3 as an initial input shape to the network. It takes 2 minutes and 14s for 50 epochs to train the model and 720 milliseconds for one epoch to test the model. Each epoch contains ten batches for training and three batches for testing. The best model gives 97.84% accuracy and 0.0873 of loss. Finally, DenseNet121 accurately predicted 9 out of 10 raw images in 6 seconds.

Table 4.13: DenseNet121 on raw version

Exp	Input	Split		Batch			Activation	Epochs	Time			Loss	Accuracy
		Train	Test	Size	Train	Test			Train	Test	Predict		
1	256x256x3	80%	20%	64	10	3	Sigmoid	100	4m 54s	270ms	7s	0.0927	0.9675
2	256x256x3	80%	20%	64	10	3	Sigmoid	500	15m 23s	270ms	5s	0.1267	0.9697
3	512x512x3	80%	20%	64	10	3	Sigmoid	100	11m 23s	909ms	6s	0.0846	0.9675
4	512x512x3	80%	20%	64	10	3	Sigmoid	500	44m 38s	906ms	7s	0.0799	0.9762
5	320x320x3	80%	20%	64	10	3	Sigmoid	100	6m 9s	399ms	7s	0.0833	0.9762
6	320x320x3	80%	20%	64	10	3	Sigmoid	500	20m 55s	399ms	7s	0.1073	0.9762
7	192x192x3	80%	20%	64	10	3	Sigmoid	100	3m 55s	180ms	7s	0.1017	0.9567
8	224x224x3	80%	20%	64	10	3	Sigmoid	100	5m 32s	222ms	7s	0.0855	0.9697
9	224x224x3	80%	20%	64	10	3	Sigmoid	500	12m 44s	225ms	7s	0.1211	0.9654
10	128x128x3	80%	20%	64	10	3	Sigmoid	100	3m 26s	117ms	7s	0.0985	0.9654
11	448x448x3	80%	20%	64	10	3	Sigmoid	500	38m 4s	723ms	6s	0.0886	0.9784
12	448x448x3	80%	20%	64	10	3	Sigmoid	50	2m 14s	720ms	6s	0.0873	0.9784

Experiment 4 using DenseNet121 on enhanced version offers 99.35% accuracy, which has the most precise result among all the experiments. The hyperparameters are similar to experiment 12 of DenseNet121 on the raw version. The only difference is the number of epochs such that experiment 12 on raw images trained in 50, and experiment 4 on enhanced version takes 500 iterations for train, as we can see in Table 4.14 with other details of the model.

Table 4.14: DenseNet121 on enhanced version

Exp	Input	Split		Batch			Activation	Epochs	Time			Loss	Accuracy
		Train	Test	Size	Train	Test			Train	Test	Predict		
1	128x128x3	80%	20%	64	10	3	Sigmoid	100	3m 15s	114ms	7s	0.079	0.9697
2	128x128x3	80%	20%	64	10	3	Sigmoid	500	7m 3s	108ms	6s	0.1062	0.9675
3	448x448x3	80%	20%	64	10	3	Sigmoid	100	9m 13s	720ms	7s	0.0438	0.9913
4	448x448x3	80%	20%	64	10	3	Sigmoid	500	38m 4s	720ms	6s	0.0277	0.9935
5	448x448x3	80%	20%	64	10	3	Sigmoid	1000	1h 13m 9s	738ms	7s	0.0280	0.9935
6	416x416x3	80%	20%	64	10	3	Sigmoid	100	8m 39s	645ms	8s	0.0442	0.9913
7	416x416x3	80%	20%	64	10	3	Sigmoid	500	25m 43s	642ms	7s	0.0292	0.9913
8	256x256x3	80%	20%	64	10	3	Sigmoid	100	4m 54s	267ms	7s	0.0528	0.9827
9	256x256x3	80%	20%	64	10	3	Sigmoid	500	15m 33s	282ms	7s	0.0458	0.9870
10	256x256x3	80%	20%	64	10	3	Sigmoid	1000	29m 20s	279ms	7s	0.0551	0.9870
11	64x64x3	80%	20%	64	10	3	Sigmoid	100	3m 21s	111ms	7s	0.1682	0.9242

4.7 Results Comparison

In this section, we have summarized all the best results based on each CNN architecture. Also, comprehensively analyzed the best models results of raw and enhanced versions, respectively.

4.7.1 Best Model - Raw/Original Images

Table 4.15 presents the most accurate evaluations for raw images of Fishes in the Wild dataset. Among these, model 5 performed more precisely. It predicts 9 out of 10 image samples with 97.84% accuracy as we discussed in 4.6. Model 3 using ResNet50 gives the lowest of 87.88% accuracy.

Table 4.15: Raw Version of FITW Dataset

Model	CNN Architecture	Experiment	Correct Predictions	Binary Accuracy
M1	VGG16	12	8 out of 10	95.38%
M2	VGG19	13	6 out of 10	95.02%
M3	ResNet50	7	7 out of 10	87.88%
M4	Xception71	9	9 out of 10	97.11%
M5	DenseNet121	12	9 out of 10	97.84%

Figure 4.3 shows training and validation accuracy rates of model 5 concerning the raw version. After 50 epochs, the training rate reaches 99% and the validation rate to 97.84%. If we look at experiment 11 in Table 4.13, we achieved the same accuracy in 500 epochs, but that increases the error rate between training and validation scores and may cause overfit of the model. So, in experiment 12, we fixed the number of epochs to 50 and gained better performance regarding the error rate.

The loss rates of training and validation of model 5 concerning the raw version have displayed in Figure 4.4.

Figure 4.5 shows the confusion matrix for each class based on the raw version. We can see that model 5 classifies 146 correctly out of 154 predictions for Rockfish and Starfish. The best model has been classified 149 accurately for Tilefish out of 154 predictions.

The classification report of Model 5 has shown in Table 4.16. We can see the different attributes such as precision, recall, and F1-score and their values.

4.7.2 Best Model - Enhanced Images

The well-performed evaluations for enhanced images of the dataset are present in Table 4.17. Model 5 gives better accuracy of 99.35%. It predicts 9 out of 10 as

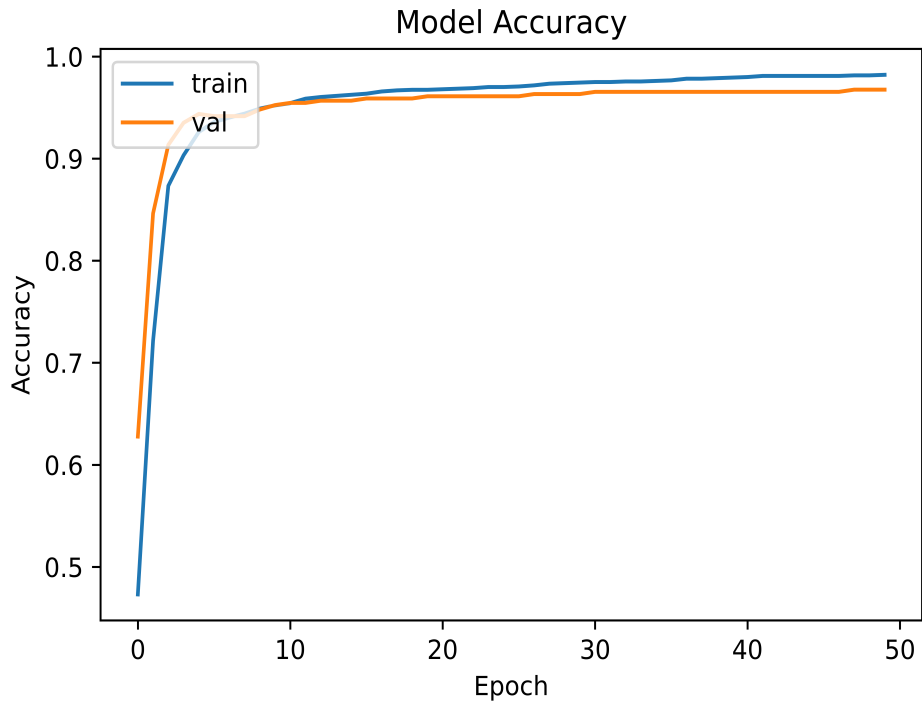


Figure 4.3: Model 5 (accuracy w.r.t raw version)

Table 4.16: Model-5 (Classification Report - Raw version)

Class	Precision	Recall	F1-score
Rockfish	0.97	0.91	0.94
Starfish	0.95	0.91	0.93
Tilefish	0.95	0.97	0.96

mentioned in 4.6. ResNet50 offers the lowest of 93.29% accuracy, as we can see in Model 3 in Table 4.17.

The training and validation accuracy rates of model 5 concerning the enhanced version have represented in Figure 4.6. As we can see, it gives 99.99% accuracy for training and 99.35% for testing after 500 epochs.

Figure 4.7 represents that there isn't enough gap between our train and test loss. So, this is the best fit model concerning the enhanced version of the dataset.

Figure 4.8 shows the confusion matrix for each class based on the enhanced

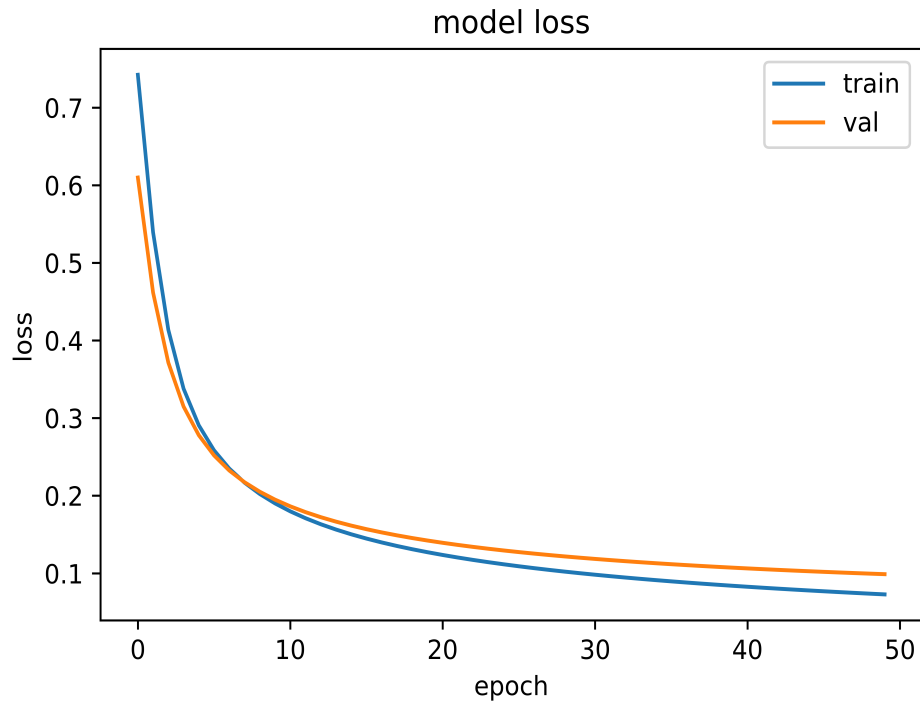


Figure 4.4: Model 5 (loss w.r.t raw version)

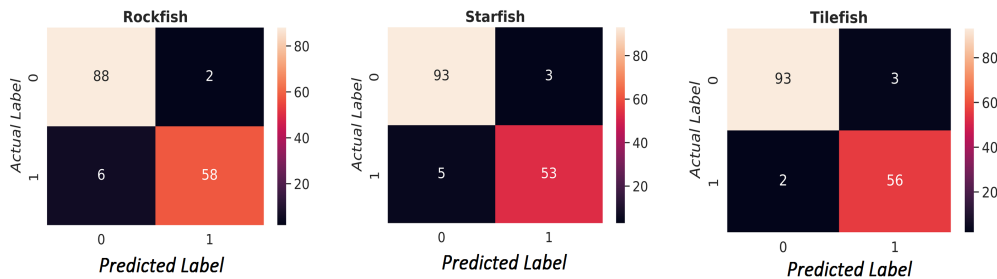


Figure 4.5: Model 5 - (Confusion Matrix w.r.t raw version)

version. We can see that model 5 classifies 151 correctly out of 154 predictions for Rockfish, 144 out of 154 for Starfish, and 152 accurately for Tilefish out of 154 predictions.

The classification report of Model 5 has shown in Table 4.18. We can see the different attributes such as precision, recall, and F1-score and their values.

4.7.3 Samples Predictions

Table 4.19 presents ten samples predictions using the best models. We can see the actual and predicted values for both enhanced and raw versions.

Table 4.17: Enhanced Version of FITW Dataset

Model	CNN Architecture	Experiment	Correct Predictions	Binary Accuracy
M1	VGG16	10	6 out of 10	96.10%
M2	VGG19	7	6 out of 10	96.10%
M3	ResNet50	16	7 out of 10	93.29%
M4	Xception71	1	7 out of 10	98.70%
M5	DenseNet121	4	9 out of 10	99.35%

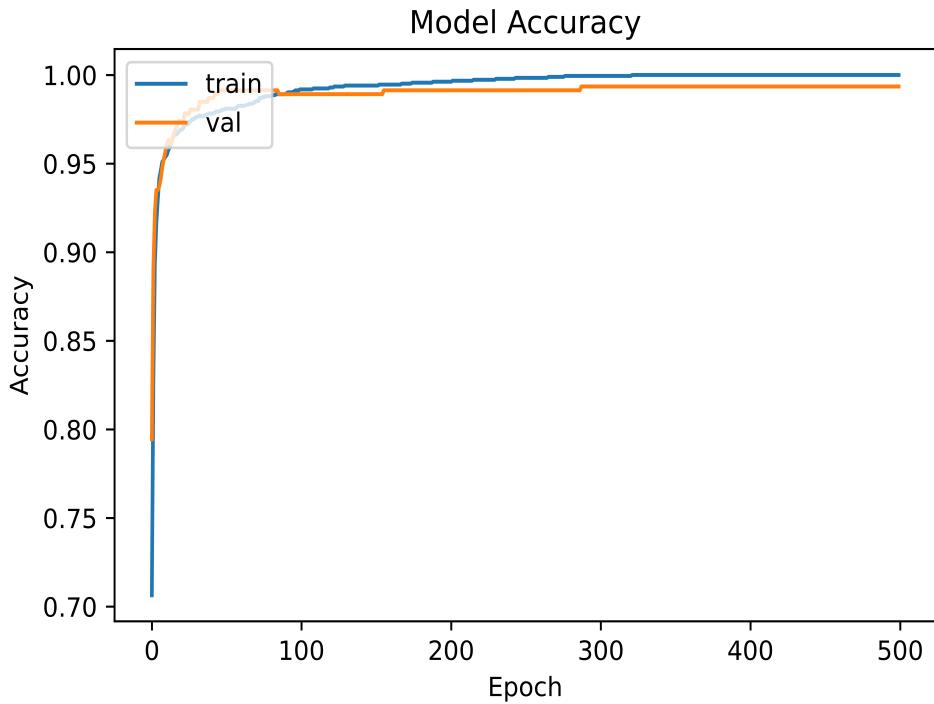


Figure 4.6: Model 5 (accuracy w.r.t enhanced version)

Table 4.18: Model-5 (Classification Report - Enhanced version)

Class	Precision	Recall	F1-score
Rockfish	1.00	0.95	0.97
Starfish	0.95	0.89	0.92
Tilefish	0.98	0.98	0.98

4.7.4 Model Summary

The complete architecture of our model for raw and the enhanced datasets has shown in Table. 4.20. The model consists of four layers such as input layer,

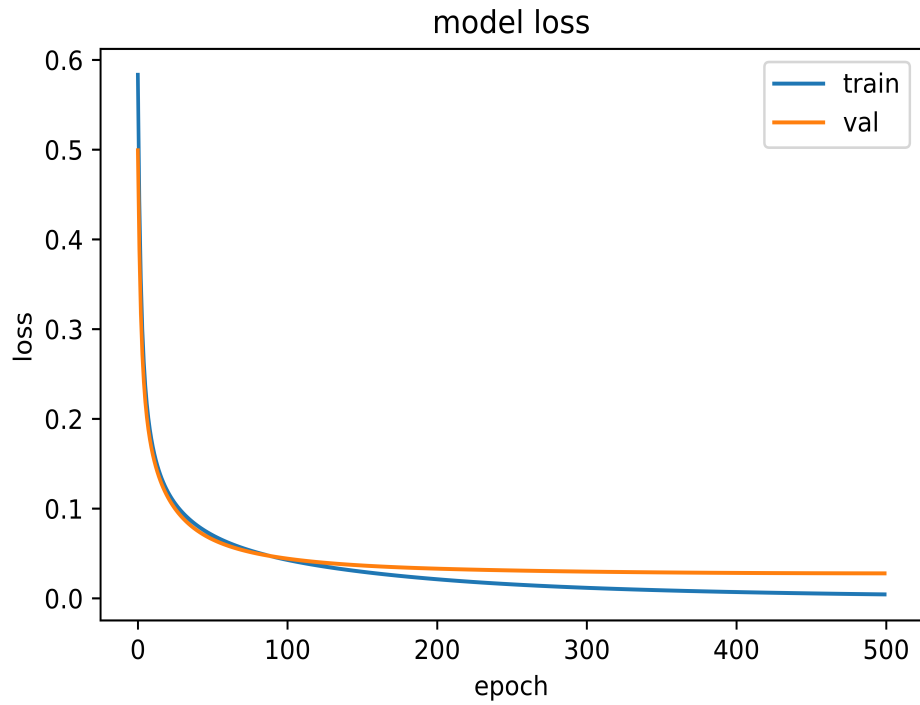


Figure 4.7: Model 5 (loss w.r.t enhanced version)

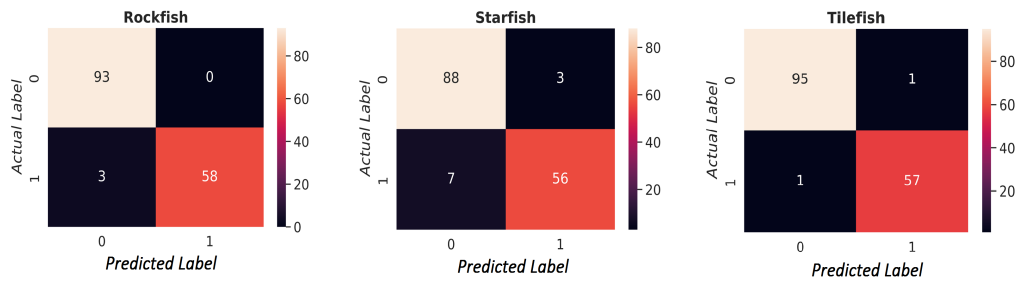


Figure 4.8: Model 5 - (Confusion Matrix w.r.t enhanced version)

functional layer, pooling layer, and output layer. We feed $448 \times 448 \times 3$ as initial input to the model that the model gives $14 \times 14 \times 1024$ shape as an input to pooling layer. The global average pooling technique has been applied in the pooling layer. In the end, the model does classify for three species as shown in the dense layer. There are 7,040,579 parameters in the proposed algorithm, of which 3,075 are in a dense layer and trainable parameters, and non-trainable parameters of 7,037,504 are in a functional (DenseNet121) layer.

Table 4.19: Ten samples predictions using best models

Raw (Model 5)		Enhanced (Model 5)	
<i>Actual Label(s)</i>	<i>Predicted</i>	<i>Actual Label(s)</i>	<i>Predicted</i>
Starfish	Starfish	Tilefish	Tilefish
Starfish, Rockfish	Starfish	Tilefish	Tilefish
Tilefish	Tilefish	Tilefish	Tilefish
Tilefish	Tilefish	Rockfish	Rockfish
Rockfish	Rockfish	Starfish, Rockfish	Starfish, Rockfish
Starfish, Rockfish	Starfish, Rockfish	Rockfish	Starfish
Tilefish	Starfish, Rockfish	Tilefish	Tilefish
Rockfish	Rockfish	Rockfish	Rockfish
Rockfish	Rockfish	Rockfish	Rockfish
Starfish	Starfish	Starfish	Starfish











Table 4.20: Model Summary

Layer	Output Shape	Parameters
Input	448x448x3	0
DenseNet121	14x14x1024	7037504
Pooling	1024	0
Dense	3	3075

4.8 Application Results

The application has been developed in Python integrated with the trained weights on the LFW dataset [27]. Firstly, we give input images from Fish Species Image Dataset [47]. Then, we use the weights of the trained CNN model to measure the probabilities of each class. We tested the application using ten sample images, and the good news is all of them gave correct results. The ten sample images consist of two images of Rockfish, Starfish, Tilefish, Rockfish+Starfish, and Rockfish+Tilefish each. The experimental results of the application with probabilities of each class as given in Table. 4.21.

Table 4.21: Application Results

Input Image	Rockfish	Starfish	Tilefish
	0.936	0.0399	0.0655
	0.948	0.00823	0.377
	0.325	0.829	0.0417
	0.384	0.936	0.00866
	0.166	0.00257	0.979
	0.479	0.00191	0.969
	0.626	0.602	0.0312
	0.509	0.36	0.106
	0.924	0.00616	0.724
	0.789	0.000711	0.974

CHAPTER 5

DISCUSSION AND CONCLUSION

We compared our proposed model with some existing methods that used different datasets for underwater image classification. The evaluation metrics accuracy values computed from the various underwater classification tasks has shown in Table 5.1. The chosen related works are based on raw/original underwater images and selected from the last six years.

Table 5.1: Comparison of proposed method with existing works (Raw/Original Images)

Method	Year	Accuracy	Reference
FRLUI	2016	75.63%	[16]
UICMLT	2019	93.00%	[20]
Few-shot	2020	36.30%	[17]
FathomNet	2020	92.80%	[23]
DUICM	2020	69.38%	[24]
Proposed method	2022	97.84%	-

Similarly, we compared our proposed model for the enhanced version datasets with some existing underwater image classification methods. The accuracy metrics computed from the various enhanced underwater classification tasks has shown in Table 5.2. We selected some of the best works from the last six years.

We compare our experiments with the latest methods for underwater image classification. In [16], Sun et al. used a model that learns the features from relatively low-resolution images by applying deep learning methods and the super-resolution approach. The researchers used LifeCLEF 2015 [84] dataset for experiments on both the original and enhanced versions. Another method named Few-shot [17] had been proposed for underwater image classification and had been implemented on three types of fish datasets that expand the training data to assure quality and quantity.

Table 5.2: Comparison of proposed method with existing works (Enhanced Images)

Method	Year	Accuracy	Reference
FRLUI	2016	77.27%	[16]
UFSC	2017	96.29%	[18]
UFSRDLT	2019	98.79%	[21]
Few-shot	2020	42.40%	[17]
ResFeats	2020	99.10%	[19]
DUICM	2020	97.37%	[24]
Proposed method	2022	99.35%	-

For each of the few-shot classes, a translation model had been used to generate high-quality images. The model [24] (DUICM) aim was to apply a convolutional neural network (CNN) on Benchmark Turbid Image Dataset [14]. The dataset had further converted into two more versions, raw and enhanced image datasets. A machine learning-based method had applied in [20], called the Bag of Features model, to tackle the mini dataset issue, and a multi-label image classification model on the original/raw underwater image datasets using ResNet50 in [23]. In [18] attempted to recognize underwater images from Fish4Knowledge dataset [44] using deep learning, and convolutional neural networks. ResFeats [19] were introduced to measure how precisely the features extracted from the various layers of a ResNet on the ImageNet dataset. Different classification techniques were used in [21] and were performed better for underwater image classification.

The proposed method performed much better than existing systems. We have used five configured CNNs such as VGG16, VGG19, ResNet50, Xception71, and DenseNet121. The DenseNet121 offered more accurate results as compared to other architectures. We configured the DenseNet121 by dividing the dataset into 80% and 20% for train and test. Then, set the batch size to 64 and fixed the sigmoid function at the output layer. After that, we assigned the initial input shape to 448x448x3. Achieved 97.84% accuracy after 50 epochs for raw image dataset and gained 99.35% accuracy after 500 iterations for the enhanced version.

We have shown that the proposed model performed well on low-resolution and high-resolution images. The DenseNet is classified well compared to other CNNs in classification problems considering underwater image problems such as blur, distinct nature, light scattering, absorption, saturation, and low-light illumination problem.

The DenseNet has reinforced feature propagation, less number of parameters, and reduces the vanishing-gradient problem [69]. Less accuracy and more wrong predictions appear in other architectures, while DenseNet predicts well in low-resolution and enhanced versions since it uses features of all complexity levels. We have applied the adaptive histogram equalization for image enhancement which reduces many underwater problems and makes images clearer to classify. The experiment on the enhanced dataset offers 1.51% more accuracy than the original version and has more precise predictions.

There are some points as given below, shows the uniqueness of this study:

- Usually, the labeled fishes in the wild dataset had used in object detection tasks. In this study, we have done fish species classification using underwater images, which can't be found in traditional studies, as we can see from Section 2.3.
- We have applied the CLAHE method on underwater images to improve illumination problem.
- Using a large dataset instead of a small dataset increased performance in both forms, such as original and enhanced.
- Using pre-trained models is a much more suitable technique than traditional ones to go deeper with layers. It also performed well in predictions, and it saves time during model creation by applying a transfer-learning approach instead of making deep models from scratch.
- By enlarging the initial input size in DenseNet from 224x224x3 to double size, i.e., 448x448x3, we have achieved the most successful results in both forms of the dataset.
- Mainly, the proposed model gives good results for unenhanced images too.

An advanced classification approach has been proposed in this study to classify underwater captured images in different conditions. The Fishes in the Wild dataset we selected for implementation. In the preprocessing step, the dataset has prepared by removing unnecessary data and annotated in a multi-label classification form. Considering class balance issues, we have limited the number of classes in the dataset to Rockfish, Starfish, and Tilefish. Then, the dataset has converted into two versions, i.e., the original and the contrasted dataset. The original version contains underwater images without enhancement. The contrasted version used CLAHE as an approach for enhancing images. We applied the transfer learning approach in the implementation part: the second strategy of pre-trained networks to our dataset,

which is most likely to our proposed problem. The already trained networks implemented in the proposed work are VGG16, VGG19, ResNet50, Xception71, and DenseNet121. The DenseNet121 gives 97.84% accuracy on the raw version, and on the enhanced version, it offers 99.35% accuracy, which has the most accurate result among all the architectures. The DenseNet model gives good results for unenhanced images since it uses features of all complexity levels. A total of 121 experiments had done on the above-mention five of CNNs. Tesla P100 GPU had used for all of the experiments. The results compared with existing methods shows that the proposed model performed better on low-resolution and high-resolution images. The main advantage of the proposed model is that it performed well while predicting fish species in the original underwater dataset. The DenseNet uses features of all complexity levels, so it overcomes many challenges while classifying underwater images such as blur images, distinct nature images, light scattering problems, absorption, saturation, and low-light illumination problems. In the future, the model should apply to different underwater datasets. Alternatively, use various data augmentation methods to deal with data scarcity and insufficient data diversity.

REFERENCES

- [1] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778.
- [2] L. Jin, H. Liang, Deep learning for underwater image recognition in small sample size situations, in: OCEANS 2017-Aberdeen, IEEE, 2017, pp. 1–4.
- [3] S. Pelletier, A. Montacir, H. Zakari, M. Akhloufi, Deep learning for marine resources classification in non-structured scenarios: training vs. transfer learning, in: 2018 IEEE Canadian Conference on Electrical & Computer Engineering (CCECE), IEEE, 2018, pp. 1–4.
- [4] J. Li, K. A. Skinner, R. M. Eustice, M. Johnson-Roberson, Watergan: Unsupervised generative network to enable real-time color correction of monocular underwater images, IEEE Robotics and Automation letters 3 (1) (2017) 387–394.
- [5] H. Lu, Y. Li, T. Uemura, H. Kim, S. Serikawa, Low illumination underwater light field images reconstruction using deep convolutional neural networks, Future Generation Computer Systems 82 (2018) 142–148.
- [6] H. Lu, Y. Li, Y. Zhang, M. Chen, S. Serikawa, H. Kim, Underwater optical image processing: a comprehensive review, Mobile networks and applications 22 (6) (2017) 1204–1211.
- [7] E. Crawford, J. Pineau, Spatially invariant unsupervised object detection with convolutional neural networks, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 33, 2019, pp. 3412–3420.
- [8] Y.-J. Cha, W. Choi, O. Büyüköztürk, Deep learning-based crack damage detection using convolutional neural networks, Computer-Aided Civil and Infrastructure Engineering 32 (5) (2017) 361–378.
- [9] S. Albawi, T. A. Mohammed, S. Al-Zawi, Understanding of a convolutional neural network, in: 2017 International Conference on Engineering and Technology (ICET), Ieee, 2017, pp. 1–6.

- [10] A. Krizhevsky, I. Sutskever, G. E. Hinton, Imagenet classification with deep convolutional neural networks, *Communications of the ACM* 60 (6) (2017) 84–90.
- [11] K. Shan, J. Guo, W. You, D. Lu, R. Bie, Automatic facial expression recognition based on a deep convolutional-neural-network structure, in: *2017 IEEE 15th International Conference on Software Engineering Research, Management and Applications (SERA)*, 2017, pp. 123–128. doi:10.1109/SERA.2017.7965717.
- [12] S. Sony, K. Dunphy, A. Sadhu, M. Capretz, A systematic review of convolutional neural network-based structural condition assessment techniques, *Engineering Structures* 226 (2021) 111347.
- [13] T. Chan, K. Jia, S. Gao, J. Lu, Z. Zeng, Y. Ma, A simple deep learning baseline for image classification? arxiv preprint, *arXiv preprint arXiv:1404.3606* 1 (3) (2014).
- [14] C. Li, C. Guo, W. Ren, R. Cong, J. Hou, S. Kwong, D. Tao, An underwater image enhancement benchmark dataset and beyond, *IEEE Transactions on Image Processing* 29 (2019) 4376–4389.
- [15] J. Wang, Y. Hu, An improved enhancement algorithm based on cnn applicable for weak contrast images, *IEEE Access* 8 (2020) 8459–8476.
- [16] X. Sun, J. Shi, J. Dong, X. Wang, Fish recognition from low-resolution underwater images, in: *2016 9th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI)*, IEEE, 2016, pp. 471–476.
- [17] Z. Guo, L. Zhang, Y. Jiang, W. Niu, Z. Gu, H. Zheng, G. Wang, B. Zheng, Few-shot fish image generation and classification, in: *Global Oceans 2020: Singapore – U.S. Gulf Coast*, 2020, pp. 1–6. doi:10.1109/IEEECONF38699.2020.9389005.
- [18] D. Rathi, S. Jain, S. Indu, Underwater fish species classification using convolutional neural network and deep learning, in: *2017 Ninth international conference on advances in pattern recognition (ICAPR)*, IEEE, 2017, pp. 1–6.
- [19] A. Mahmood, M. Bennamoun, S. An, F. Sohel, F. Boussaid, Resfeats: Residual network based features for underwater image classification, *Image and Vision Computing* 93 (2020) 103811.
- [20] M. V. Raj, S. S. Murugan, Underwater image classification using machine learning technique, in: *2019 International Symposium on Ocean Technology (SYM-POL)*, 2019, pp. 166–173. doi:10.1109/SYMPOL48207.2019.9005299.

- [21] B. V. Deep, R. Dash, Underwater fish species recognition using deep learning techniques, in: 2019 6th International Conference on Signal Processing and Integrated Networks (SPIN), 2019, pp. 665–669. doi:10.1109/SPIN.2019.8711657.
- [22] E. Silva Vaz, E. F. de Toledo, P. L. Drews, Underwater depth estimation based on water classification using monocular image, in: 2020 Latin American Robotics Symposium (LARS), 2020 Brazilian Symposium on Robotics (SBR) and 2020 Workshop on Robotics in Education (WRE), 2020, pp. 1–6. doi:10.1109/LARS/SBR/WRE51543.2020.9307103.
- [23] O. Boulais, B. Woodward, B. Schlining, L. Lundsten, K. Barnard, K. C. Bell, K. Katija, Fathomnet: An underwater image training database for ocean exploration and discovery, arXiv preprint arXiv:2007.00114 (2020).
- [24] M. Aridoss, C. Dhasarathan, A. Dumka, J. Loganathan, Duicm deep underwater image classification model using convolutional neural networks, International Journal of Grid and High Performance Computing (IJGHPC) 12 (3) (2020) 88–100.
- [25] D. P. Williams, Transfer learning with sas-image convolutional neural networks for improved underwater target classification, in: IGARSS 2019 - 2019 IEEE International Geoscience and Remote Sensing Symposium, 2019, pp. 78–81. doi:10.1109/IGARSS.2019.8898611.
- [26] A. Duarte, F. Codevilla, J. D. O. Gaya, S. S. Botelho, A dataset to evaluate underwater image restoration methods, in: OCEANS 2016-Shanghai, IEEE, 2016, pp. 1–6.
- [27] N. F. N. M. F. Service), Labeled fishes in the wild dataset, available from: <https://swfscdata.nmfs.noaa.gov/labeled-fishes-in-the-wild/>.
- [28] Z. Liang, X. Ding, Y. Wang, X. Yan, X. Fu, Gudcp: Generalization of underwater dark channel prior for underwater image restoration, IEEE Transactions on Circuits and Systems for Video Technology (2021) 1–1doi:10.1109/TCSVT.2021.3114230.
- [29] J. Zhou, D. Zhang, P. Zou, W. Zhang, W. Zhang, Retinex-based laplacian pyramid method for image defogging, IEEE Access 7 (2019) 122459–122472. doi:10.1109/ACCESS.2019.2934981.
- [30] R. Caballero, A. Berbey-Alvarez, Underwater image enhancement using dark channel prior and image opacity, in: 2019 7th International Engineering, Sciences and Technology Conference (IESTEC), 2019, pp. 556–561. doi:10.1109/IESTEC46403.2019.00105.

- [31] H. Feng, L. Xu, X. Yin, Z. Chen, Underwater salient object detection based on red channel correction, in: 2021 IEEE 2nd International Conference on Big Data, Artificial Intelligence and Internet of Things Engineering (ICBAIE), 2021, pp. 446–449. doi:10.1109/ICBAIE52039.2021.9390003.
- [32] L. Bai, W. Zhang, X. Pan, C. Zhao, Underwater image enhancement based on global and local equalization of histogram and dual-image multi-scale fusion, *IEEE Access* 8 (2020) 128973–128990.
- [33] S. S. Subah, M. A. Islam, M. M. Islam, Underwater image enhancement based on fusion technique via color correction and illumination adjustment, in: 2019 1st International Conference on Advances in Science, Engineering and Robotics Technology (ICASERT), 2019, pp. 1–5. doi:10.1109/ICASERT.2019.8934573.
- [34] V. Jakhetiya, K. Gu, S. P. Jaiswal, T. Singhal, Z. Xia, Kernel-ridge regression-based quality measure and enhancement of three-dimensional-synthesized images, *IEEE Transactions on Industrial Electronics* 68 (1) (2021) 423–433. doi:10.1109/TIE.2020.2965469.
- [35] C.-Y. Li, J.-C. Guo, R.-M. Cong, Y.-W. Pang, B. Wang, Underwater image enhancement by dehazing with minimum information loss and histogram distribution prior, *IEEE Transactions on Image Processing* 25 (12) (2016) 5664–5677. doi:10.1109/TIP.2016.2612882.
- [36] Y. Ueki, M. Ikehara, Underwater image enhancement based on the iteration of a generalization of dark channel prior, in: 2019 IEEE Visual Communications and Image Processing (VCIP), 2019, pp. 1–4. doi:10.1109/VCIP47243.2019.8965726.
- [37] S. H. Majeed, N. A. M. Isa, Adaptive entropy index histogram equalization for poor contrast images, *IEEE Access* 9 (2021) 6402–6437. doi:10.1109/ACCESS.2020.3048148.
- [38] Y. Chang, C. Jung, P. Ke, H. Song, J. Hwang, Automatic contrast-limited adaptive histogram equalization with dual gamma correction, *Ieee Access* 6 (2018) 11782–11792.
- [39] S. Duraibi, W. Alhamdani, F. T. Sheldon, Voice feature learning using convolutional neural networks designed to avoid replay attacks, in: 2020 IEEE Symposium Series on Computational Intelligence (SSCI), IEEE, 2020, pp. 1845–1851.
- [40] W. Aly, S. Aly, S. Almotairi, User-independent american sign language alphabet recognition based on depth image and pcanet features, *IEEE Access* 7 (2019) 123138–123150.

- [41] X. Qi, X. Wu, Y. Ji, X. Wang, H. Li, Research on classification of power load data based on libsvm, in: 2019 11th International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC), Vol. 2, 2019, pp. 158–162. doi:10.1109/IHMSC.2019.10133.
- [42] A. Chetouani, T. Debrouetelle, S. Treuillet, M. Exbrayat, S. Jesset, Classification of ceramic shards based on convolutional neural network, in: 2018 25th IEEE International Conference on Image Processing (ICIP), IEEE, 2018, pp. 1038–1042.
- [43] S. Luan, C. Chen, B. Zhang, J. Han, J. Liu, Gabor convolutional networks, IEEE Transactions on Image Processing 27 (9) (2018) 4357–4366.
- [44] J. Nie, L. Huang, W. Zhang, G. Wei, Z. Wei, Deep feature ranking for person re-identification, IEEE Access 7 (2019) 15007–15017.
- [45] G. Cutter, K. Stierhoff, J. Zeng, Automated detection of rockfish in unconstrained underwater videos using haar cascades and a new image dataset: Labeled fishes in the wild, in: 2015 IEEE Winter Applications and Computer Vision Workshops, 2015, pp. 57–62. doi:10.1109/WACVW.2015.11.
- [46] M. Sung, S.-C. Yu, Y. Girdhar, Vision based real-time fish detection using convolutional neural network, in: OCEANS 2017 - Aberdeen, 2017, pp. 1–6. doi:10.1109/OCEANSE.2017.8084889.
- [47] S. Srinivasan, Fish species image data, <https://www.kaggle.com/datasets/sripaadsrinivasan/fish-species-image-data> (2021).
- [48] J. Ma, X. Fan, S. X. Yang, X. Zhang, X. Zhu, Contrast limited adaptive histogram equalization-based fusion in yiq and hsi color spaces for underwater image enhancement, International Journal of Pattern Recognition and Artificial Intelligence 32 (07) (2018) 1854018.
- [49] V. Stimper, S. Bauer, R. Ernstorfer, B. Schölkopf, R. P. Xian, Multidimensional contrast limited adaptive histogram equalization, IEEE Access 7 (2019) 165437–165447. doi:10.1109/ACCESS.2019.2952899.
- [50] A. Momeni Pour, H. Seyedarabi, S. H. Abbasi Jahromi, A. Javadzadeh, Automatic detection and monitoring of diabetic retinopathy using efficient convolutional neural networks and contrast limited adaptive histogram equalization, IEEE Access 8 (2020) 136668–136673. doi:10.1109/ACCESS.2020.3005044.
- [51] P. Singh, R. Mukundan, R. De Ryke, Feature enhancement in medical ultrasound videos using contrast-limited adaptive histogram equalization, Journal of Digital Imaging 33 (1) (2020) 273–285.

- [52] S. Albawi, T. A. Mohammed, S. Al-Zawi, Understanding of a convolutional neural network, in: 2017 International Conference on Engineering and Technology (ICET), 2017, pp. 1–6. doi:10.1109/ICEngTechnol.2017.8308186.
- [53] F. Sultana, A. Sufian, P. Dutta, Advancements in image classification using convolutional neural network, in: 2018 Fourth International Conference on Research in Computational Intelligence and Communication Networks (ICRCICN), 2018, pp. 122–129. doi:10.1109/ICRCICN.2018.8718718.
- [54] K.-H. Chan, G. Pau, S.-K. Im, Chebyshev pooling: An alternative layer for the pooling of cnns-based classifier, in: 2021 IEEE 4th International Conference on Computer and Communication Engineering Technology (CCET), 2021, pp. 106–110. doi:10.1109/CCET52649.2021.9544405.
- [55] K. Liu, G. Kang, N. Zhang, B. Hou, Breast cancer classification based on fully-connected layer first convolutional neural networks, *IEEE Access* 6 (2018) 23722–23732. doi:10.1109/ACCESS.2018.2817593.
- [56] Y. Gao, K. M. Mosalam, Deep transfer learning for image-based structural damage recognition, *Computer-Aided Civil and Infrastructure Engineering* 33 (9) (2018) 748–768.
- [57] S. Shao, S. McAleer, R. Yan, P. Baldi, Highly accurate machine fault diagnosis using deep transfer learning, *IEEE Transactions on Industrial Informatics* 15 (4) (2018) 2446–2455.
- [58] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, Q. He, A comprehensive survey on transfer learning, *Proceedings of the IEEE* 109 (1) (2020) 43–76.
- [59] P. Marcelino, Transfer learning from pre-trained models, *Towards Data Science* (2018).
- [60] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, *Proceedings of the IEEE* 86 (11) (1998) 2278–2324.
- [61] A. Krizhevsky, I. Sutskever, G. E. Hinton, Imagenet classification with deep convolutional neural networks, *Advances in neural information processing systems* 25 (2012) 1097–1105.
- [62] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, *arXiv preprint arXiv:1409.1556* (2014).

- [63] F. Chollet, Xception: Deep learning with depthwise separable convolutions, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 1251–1258.
- [64] G. Huang, Z. Liu, L. Van Der Maaten, K. Q. Weinberger, Densely connected convolutional networks, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 4700–4708.
- [65] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, Z. Wojna, Rethinking the inception architecture for computer vision, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 2818–2826.
- [66] X. Xia, C. Xu, B. Nan, Inception-v3 for flower classification, in: 2017 2nd International Conference on Image, Vision and Computing (ICIVC), IEEE, 2017, pp. 783–787.
- [67] M. ul Hassan, Vgg16-convolutional network for classification and detection, en línea.[consulta: 10 abril 2019]. Disponible en: <https://neurohive.io/en/popular-networks/vgg16> (2018).
- [68] X. Han, R. Jin, A small sample image recognition method based on resnet and transfer learning, in: 2020 5th International Conference on Computational Intelligence and Applications (ICCIA), 2020, pp. 76–81. doi:10.1109/ICCIA49625.2020.00022.
- [69] Y. Zhu, S. Newsam, Densenet for dense flow, in: 2017 IEEE international conference on image processing (ICIP), IEEE, 2017, pp. 790–794.
- [70] S. Chand, et al., A comparative study of breast cancer tumor classification by classical machine learning methods and deep learning method, Machine Vision and Applications 31 (6) (2020) 1–10.
- [71] M. Karakaya, How to solve classification problems in deep learning with tensorflow & keras?, <https://medium.com/deep-learning-with-keras/how-to-solve-classification-problems-in-deep-learning-with-tensorflow-keras-6e39c5b09501> (2021).
- [72] T. P. Lillicrap, A. Santoro, L. Marris, C. J. Akerman, G. Hinton, Backpropagation and the brain, Nature Reviews Neuroscience 21 (6) (2020) 335–346.
- [73] Y. Wang, Y. Li, Y. Song, X. Rong, The influence of the activation function in a convolution neural network model of facial expression recognition, Applied Sciences 10 (5) (2020) 1897.

- [74] I. Kouretas, V. Paliouras, Simplified hardware implementation of the softmax activation function, in: 2019 8th International Conference on Modern Circuits and Systems Technologies (MOCASST), 2019, pp. 1–4. doi:10.1109/MOCASST.2019.8741677.
- [75] G. Lin, W. Shen, Research on convolutional neural network based on improved relu piecewise activation function, *Procedia computer science* 131 (2018) 977–984.
- [76] K. Eckle, J. Schmidt-Hieber, A comparison of deep networks with relu activation function and linear spline-type methods, *Neural Networks* 110 (2019) 232–242.
- [77] U. Ruby, V. Yendapalli, Binary cross entropy with deep learning technique for image classification, *International Journal of Advanced Trends in Computer Science and Engineering* 9 (10) (2020).
- [78] E. Gordon-Rodriguez, G. Loaiza-Ganem, G. Pleiss, J. P. Cunningham, Uses and abuses of the cross-entropy loss: case studies in modern deep learning, *Proceedings of Machine Learning Research* 137 (2020).
- [79] P. Singh, N. Singh, K. K. Singh, A. Singh, Diagnosing of disease using machine learning, in: *Machine Learning and the Internet of Medical Things in Healthcare*, Elsevier, 2021, pp. 89–111.
- [80] A. Kulkarni, D. Chong, F. A. Batarseh, Foundations of data imbalance and solutions for a data democracy, in: *Data Democracy*, Elsevier, 2020, pp. 83–106.
- [81] M. Hossin, M. N. Sulaiman, A review on evaluation metrics for data classification evaluations, *International journal of data mining & knowledge management process* 5 (2) (2015) 1.
- [82] T. C. v2.7.0, Tensorflow keras metrics binary accuracy, https://www.tensorflow.org/api_docs/python/tf/keras/metrics/binary_accuracy/(2021).
- [83] Roboflow, 2021 roboflow, inc. all rights reserved., <https://roboflow.com/> (2021).
- [84] A. Joly, H. Goëau, H. Glotin, C. Spampinato, P. Bonnet, W.-P. Vellinga, R. Planqué, A. Rauber, S. Palazzo, B. Fisher, et al., Lifeclef 2015: multimedia life species identification challenges, in: *International Conference of the Cross-Language Evaluation Forum for European Languages*, Springer, 2015, pp. 462–483.

APPENDIX A

1- Connecting to Google Colab

```
#cell
from google.colab import drive
drive.mount('/content/gdrive')
```

```
#cell
!nvidia-smi
```

```
#cell
!ln -s /content/gdrive/My\ Drive/ /mydrive
```

```
#cell
%cd /mydrive/Multi-label\ Classification/FITW
```

2- Preprocessing

```
#cell
import warnings
warnings.filterwarnings("ignore")
import tensorflow as tf
import numpy as np
import tensorflow_datasets as tfds
from tensorflow import keras
import matplotlib.pyplot as plt
import os
import PIL
import PIL.Image
import pandas as pd
import pathlib, datetime
```

```
#cell
df=pd.read_csv("Raw.csv")
df.columns = df.columns.str.replace(' ','')
```

```

df.head()

#cell
#df[['filename', 'Rockfish', 'Starfish', 'Tilefish']].groupby(['Rockfish',
'Starfish', 'Tilefish']).agg(['count'])
LABELS=["Rockfish", "Starfish", "Tilefish"]
data_dir = pathlib.Path("Raw")
filenames = list(data_dir.glob('*.*jpg'))
fnames=[]
for fname in filenames:
    fnames.append(str(fname))
print(len(fnames))
ds_size= len(fnames)
print("Number of images in folders: ", ds_size)
number_of_selected_samples=2000
filelist_ds = tf.data.Dataset.from_tensor_slices(fnames[:number_of_selected_samples])
ds_size= filelist_ds.cardinality().numpy()
print("Number of selected samples for dataset: ", ds_size)

#cell
def get_label(file_path):
    parts = tf.strings.split(file_path, '/')
    file_name= parts[-1]
    labels= df[df["filename"]==file_name][LABELS].to_numpy().squeeze()
    return tf.convert_to_tensor(labels)

#cell
IMG_WIDTH, IMG_HEIGHT = 448 , 448
def decode_img(img):
    #color images
    img = tf.image.decode_jpeg(img, channels=3)
    #convert unit8 tensor to floats in the [0,1]range
    img = tf.image.convert_image_dtype(img, tf.float32)
    #resize
    return tf.image.resize(img, [IMG_WIDTH, IMG_HEIGHT])

#cell
def combine_images_labels(file_path: tf.Tensor):

```



```

    label = get_label(file_path)
    img = tf.io.read_file(file_path)
    img = decode_img(img)
    return img, label
train_ratio = 0.80
ds_train=filelist_ds.take(ds_size*train_ratio)
ds_test=filelist_ds.skip(ds_size*train_ratio)
BATCH_SIZE= 64

#cell
ds_train=ds_train.map(lambda x: tf.py_function(func=combine_images_labels,
        inp=[x], Tout=(tf.float32,tf.int64)),
        num_parallel_calls=tf.data.AUTOTUNE,
        deterministic=False)
ds_test= ds_test.map(lambda x: tf.py_function(func=combine_images_labels,
        inp=[x], Tout=(tf.float32,tf.int64)),
        num_parallel_calls=tf.data.AUTOTUNE,
        deterministic=False)

#cell
def covert_onehot_string_labels(label_string,label_onehot):
    labels=[]
    for i, label in enumerate(label_string):
        if label_onehot[i]:
            labels.append(label)
    if len(labels)==0:
        labels.append("NONE")
    return labels

#cell
def show_samples(dataset):
    fig=plt.figure(figsize=(16, 16))
    columns = 3
    rows = 3
    print(columns*rows,"samples from the dataset")
    i=1
    for a,b in dataset.take(columns*rows):
        fig.add_subplot(rows, columns, i)
        plt.imshow(np.squeeze(a))

```

```

plt.title("image shape:"+ str(a.shape)+" (" +str(b.numpy()) +") "+
          str(covert_onehot_string_labels(LABELS,b.numpy())))
i=i+1
plt.show()
show_samples(ds_test)

#cell
#buffer_size = ds_train_resize_scale.cardinality().numpy()/10
#ds_resize_scale_batched=ds_raw.repeat(3).shuffle(buffer_size=
buffer_size).batch(64, )

ds_train_batched=ds_train.batch(BATCH_SIZE).cache().prefetch(tf.data.
experimental.AUTOTUNE)
ds_test_batched=ds_test.batch(BATCH_SIZE).cache().prefetch(tf.data.
experimental.AUTOTUNE)

print("Number of batches in train: ", ds_train_batched.cardinality().numpy())
print("Number of batches in test: ", ds_test_batched.cardinality().numpy())

```

3- Create a Keras CNN model by using Transfer Learning

#We run each pretrained CNN architecture individually for each flow of code from the start.

#1- VGG16

```

base_model = keras.applications.VGG16(
    weights='imagenet', # Load weights pre-trained on ImageNet.
    input_shape=(448, 448, 3), # expects min 32 x 32
    include_top=False) # Do not include the ImageNet classifier at the top.
base_model.trainable = False

```

#2- VGG19

```

base_model = keras.applications.VGG19(
    weights='imagenet', # Load weights pre-trained on ImageNet.
    input_shape=(448, 448, 3), # expects min 32 x 32
    include_top=False) # Do not include the ImageNet classifier at the top.
base_model.trainable = False

```

```

#3- RESNET50
base_model = keras.applications.ResNet50(
    weights='imagenet', # Load weights pre-trained on ImageNet.
    input_shape=(448, 448, 3), # expects min 32 x 32
    include_top=False) # Do not include the ImageNet classifier at the top.
base_model.trainable = False

#4- XCEPTION71
base_model = keras.applications.Xception71(
    weights='imagenet', # Load weights pre-trained on ImageNet.
    input_shape=(448, 448, 3), # expects min 32 x 32
    include_top=False) # Do not include the ImageNet classifier at the top.
base_model.trainable = False

#5- DENSENET121
base_model = keras.applications.DenseNet121(
    weights='imagenet', # Load weights pre-trained on ImageNet.
    input_shape=(448, 448, 3), # expects min 32 x 32
    include_top=False) # Do not include the ImageNet classifier at the top.
base_model.trainable = False

#cell
number_of_classes = 3

#cell
inputs = keras.Input(shape=(448 , 448 , 3))
x = base_model(inputs, training=False)
x = keras.layers.GlobalAveragePooling2D()(x)
initializer = tf.keras.initializers.GlorotUniform(seed=42)

activation = tf.keras.activations.sigmoid #None
# tf.keras.activations.sigmoid or softmax

outputs = keras.layers.Dense(number_of_classes,
                              kernel_initializer=initializer,
                              activation=activation)(x)
model = keras.Model(inputs, outputs)

```

```

#cell
model.compile(optimizer=keras.optimizers.Adam(),
              loss=keras.losses.BinaryCrossentropy(), # default from_logits=False
              metrics=[keras.metrics.BinaryAccuracy()])

#cell
history=model.fit(ds_train_batched, validation_data=ds_test_batched, epochs=50)

#Evaluate the model
ds= ds_test_batched
print("Test Accuracy: ", model.evaluate(ds)[1])

#Predictions
ds=ds_test
predictions= model.predict(ds.batch(batch_size=767).take(1))
print("A sample output from the last layer (model) ", predictions[0])
y=[]
print("10 Sample predictions:")
i = 0
for (pred,(a,b)) in zip(predictions,ds.take(767)):

    pred[pred>0.5]=1
    pred[pred<=0.5]=0
    print("predicted: " ,pred, str(covert_onehot_string_labels(LABELS, pred)),
          "Actual Label: (" +str(covert_onehot_string_labels(LABELS,b.numpy())) +)")
    y.append(b.numpy())
    i=i+1
    print(i)

from sklearn.metrics import classification_report

print(classification_report(y, predictions, target_names=LABELS))

```

4- Plotting Graphs

```

#ACCURACY

plt.plot(history.history['binary_accuracy'])

```

```

plt.plot(history.history['val_binary_accuracy'])
plt.title('Model Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend(['train', 'val'], loc='upper left')
plt.savefig('raw/Accuracy_Exp12.png', dpi=1200)
plt.show()

#LOSS GRAPH
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'val'], loc='upper right')
plt.savefig('raw/Loss_Exp12.png', dpi=1200)
plt.show()

#Confusion Matrix

import seaborn as sn
import pandas as pd
import matplotlib.pyplot as plt
array = [[88, 2],[ 6, 58]]
#raw array = [ [[88, 2],[ 6, 58]], [[93, 3],[ 5, 53]], [[93, 3],[ 2, 56]] ]
#enh array([[93, 0], [ 3, 58]], [[88, 3], [ 7, 56]], [[95, 1], [ 1, 57]])
df_cm = pd.DataFrame(array, range(2), range(2))
# plt.figure(figsize=(10,7))
sn.set(font_scale=1.4) # for label size
sn.heatmap(df_cm, annot=True, annot_kws={"size": 16}) # font size
plt.title('Rockfish', fontweight="bold")
plt.ylabel('Actual Label', fontstyle="italic")
plt.xlabel('Predicted Label', fontstyle="italic")
plt.savefig('CM2/R-R.pdf')
plt.show()
#plt.legend(['train', 'val'], loc='upper right')
#plt.save()

```

5- APPLICATION

```
from keras.preprocessing import image

img = image.load_img('APP2/T2.jpg', target_size=(448,448,3))

img = image.img_to_array(img)
img = img/255.
plt.imshow(img)
img = np.expand_dims(img, axis=0)

classes = LABELS #Get array of all classes
proba = model.predict(img) #Get probabilities for each class
sorted_categories = np.argsort(proba[0])[:-4:-1]
#Get class names for top 10 categories

#Print classes and corresponding probabilities
for i in range(3):
    print("{}".format(classes[sorted_categories[i]])+"
    ( {:.3} )".format(proba[0][sorted_categories[i]]))
```

Muhammad Imran

ORIGINALITY REPORT

9%

SIMILARITY INDEX

4%

INTERNET SOURCES

6%

PUBLICATIONS

2%

STUDENT PAPERS

PRIMARY SOURCES

1

deepai.org

Internet Source

1%

2

Submitted to Higher Education Commission
Pakistan

Student Paper

1%

3

Manimaran Aridoss, Chandramohan
Dhasarathan, Ankur Dumka, Jayakumar
Loganathan. "DUICM Deep Underwater Image
Classification Model using Convolutional
Neural Networks", International Journal of
Grid and High Performance Computing, 2020

Publication

<1%

4

Cutter, George, Kevin Stierhoff, and Jiaming
Zeng. "Automated Detection of Rockfish in
Unconstrained Underwater Videos Using Haar
Cascades and a New Image Dataset: Labeled
Fishes in the Wild", 2015 IEEE Winter
Applications and Computer Vision Workshops,
2015.

Publication

<1%

5

David P. Williams. "Transfer Learning with SAS-Image Convolutional Neural Networks for Improved Underwater Target Classification", IGARSS 2019 - 2019 IEEE International Geoscience and Remote Sensing Symposium, 2019

Publication

<1 %

6

Dhruv Rathi, Sushant Jain, S. Indu. "Underwater Fish Species Classification using Convolutional Neural Network and Deep Learning", 2017 Ninth International Conference on Advances in Pattern Recognition (ICAPR), 2017

Publication

<1 %

7

Minsung Sung, Son-Cheol Yu, Yogesh Girdhar. "Vision based real-time fish detection using convolutional neural network", OCEANS 2017 - Aberdeen, 2017

Publication

<1 %

8

Submitted to University of Westminster

Student Paper

<1 %

9

arxiv.org

Internet Source

<1 %

10

Muhammad Babar, Fahim Arif, Mian Ahmad Jan, Zhiyuan Tan, Fazlullah Khan. "Urban data management system: Towards Big Data analytics for Internet of Things based smart

<1 %

urban environment using customized Hadoop", Future Generation Computer Systems, 2019

Publication

11

Geovanni Figueroa-Mata, Erick Mata-Montero. "Using a Convolutional Siamese Network for Image-Based Plant Species Identification with Small Datasets", Biomimetics, 2020

Publication

<1 %

12

Lin Li, Eric Rigall, Junyu Dong, Geng Chen. "Chapter 12 MAS3K: An Open Dataset for Marine Animal Segmentation", Springer Science and Business Media LLC, 2021

Publication

<1 %

13

Submitted to University of Sydney

Student Paper

<1 %

14

"Advances in Computational Intelligence Systems", Springer Science and Business Media LLC, 2019

Publication

<1 %

15

Farhana Sultana, Abu Sufian, Paramartha Dutta. "Advancements in Image Classification using Convolutional Neural Network", 2018 Fourth International Conference on Research in Computational Intelligence and Communication Networks (ICRCICN), 2018

Publication

<1 %

16

Ling Yang, Yeqi Liu, Huihui Yu, Xiaomin Fang, Lihua Song, Daoliang Li, Yingyi Chen.

"Computer Vision Models in Intelligent Aquaculture with Emphasis on Fish Detection and Behavior Analysis: A Review", Archives of Computational Methods in Engineering, 2020

Publication

<1 %

17

repositori.uji.es

Internet Source

<1 %

18

"Intelligent Computing and Applications", Springer Science and Business Media LLC, 2021

Publication

<1 %

19

Submitted to University of Technology

Student Paper

<1 %

20

blog.knoldus.com

Internet Source

<1 %

21

link.springer.com

Internet Source

<1 %

22

researchoutput.csu.edu.au

Internet Source

<1 %

23

towardsdatascience.com

Internet Source

<1 %

24

Submitted to Covenant University

Student Paper

<1 %

25	Submitted to Coventry University Student Paper	<1 %
26	gisfiles.wm.edu Internet Source	<1 %
27	Submitted to Universiti Teknologi MARA Student Paper	<1 %
28	Akshay Rai, Mira Mitra. "Lamb wave based damage detection in metallic plates using multi-headed 1-dimensional convolutional neural network", Smart Materials and Structures, 2021 Publication	<1 %
29	Communications in Computer and Information Science, 2011. Publication	<1 %
30	Mathieu Juncker, Ismail Khriiss, Jean Brousseau, Steven Pigeon, Alexis Darisse, Billy Lapointe. "A Deep Learning-Based Approach for Quality Control and Defect Detection for Industrial Bagging Systems", 2020 IEEE 19th International Conference on Cognitive Informatics & Cognitive Computing (ICCI*CC), 2020 Publication	<1 %
31	docs.neu.edu.tr Internet Source	<1 %

32	www.coursehero.com Internet Source	<1 %
33	Submitted to University of Northumbria at Newcastle Student Paper	<1 %
34	studentsrepo.um.edu.my Internet Source	<1 %
35	eprints.uthm.edu.my Internet Source	<1 %
36	ijdmta.com Internet Source	<1 %
37	www.mdpi.com Internet Source	<1 %
38	olicallaghan.com Internet Source	<1 %
39	vubirelec.be Internet Source	<1 %
40	"Advances in Simulation and Digital Human Modeling", Springer Science and Business Media LLC, 2021 Publication	<1 %
41	Jia-Hong Lee, Mei-Yi Wu, Zhi-Cheng Guo. "A tank fish recognition and tracking system using computer vision techniques", 2010 3rd	<1 %

International Conference on Computer Science and Information Technology, 2010

Publication

-
- | | | |
|----|---|------|
| 42 | Koji Kashihara. "Deep convolutional neural networks improve vein image quality", 2016 IEEE 17th International Symposium on Computational Intelligence and Informatics (CINTI), 2016
Publication | <1 % |
| 43 | eprints.utm.my
Internet Source | <1 % |
| 44 | www.itc.nl
Internet Source | <1 % |
| 45 | www.lib.umd.edu
Internet Source | <1 % |
| 46 | A.F. Laine, S. Schuler, Jian Fan, W. Huda. "Mammographic feature enhancement by multiscale analysis", IEEE Transactions on Medical Imaging, 1994
Publication | <1 % |
| 47 | Asif Iqbal Khan, Junaid Latief Shah, Mohammad Mudasir Bhat. "CoroNet: A deep neural network for detection and diagnosis of COVID-19 from chest x-ray images", Computer Methods and Programs in Biomedicine, 2020
Publication | <1 % |
-

48

Prachi Mathur, Kanasani Monica, Badal Soni. "Improved Fusion-based Technique for Underwater Image Enhancement", 2018 4th International Conference on Computing Communication and Automation (ICCCA), 2018

Publication

<1 %

49

Pudumalar S, Suriya K S, Rohini K. "chapter 8 Data Classification and Prediction", IGI Global, 2018

Publication

<1 %

50

downloads.hindawi.com

Internet Source

<1 %

51

ebin.pub

Internet Source

<1 %

52

etheses.whiterose.ac.uk

Internet Source

<1 %

53

www.depositonce.tu-berlin.de

Internet Source

<1 %

54

www.nero.noaa.gov

Internet Source

<1 %

55

"Recent Advances in Information and Communication Technology 2017", Springer Science and Business Media LLC, 2018

Publication

<1 %

Exclude quotes Off

Exclude matches Off

Exclude bibliography Off

Muhammad Imran

PAGE 1

PAGE 2

PAGE 3

PAGE 4

PAGE 5

PAGE 6

PAGE 7

PAGE 8

PAGE 9

PAGE 10

PAGE 11

PAGE 12

PAGE 13

PAGE 14

PAGE 15

PAGE 16

PAGE 17

PAGE 18

PAGE 19

PAGE 20

PAGE 21

PAGE 22

PAGE 23

PAGE 24

PAGE 25

PAGE 26

PAGE 27

PAGE 28

PAGE 29

PAGE 30

PAGE 31

PAGE 32

PAGE 33

PAGE 34

PAGE 35

PAGE 36

PAGE 37

PAGE 38

PAGE 39

PAGE 40

PAGE 41

PAGE 42

PAGE 43

PAGE 44

PAGE 45

PAGE 46

PAGE 47

PAGE 48

PAGE 49

PAGE 50

PAGE 51
