HAWASSA UNIVERSITY

INSTITUTE OF TECHNOLOGY

SCHOOL OF GRADUATE STUDIES

FACULTY OF INFORMATICS

AUTOMATIC FISH SPECIES IDENTIFICATION USING DEEP LEARNING TECHNIQUE

By

HABTAMUA ZERIHUN TSEGAYE

HAWASSA UNIVERSITY, HAWASSA, ETHIOPIA

NOVEMBER, 2023

AUTOMATIC FISH SPECIES IDENTIFICATION USING DEEP LEARNING TECHNIQUE

By

HABTAMUA ZERIHUN

A THESIS SUBMITTED TO THE SCHOOL OF GRADUATE STUDIES OF HAWASSA UNIVERSITY INSTITUTE OF TECHNOLOGY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF SCIENCE IN COMPUTER SCIENCE

HAWASSA, ETHIOPIA

Program: MSc. Computer science

Major Advisor: Degif Teka (PhD)

Co-Advisor: Wogene Alemayehu (MSc)

NOVEMBER, 2023

HAWASSA UNIVERSITY

INSTITUTE OF TECHNOLOGY

SCHOOL OF GRADUATE STUDIES FACULTY OF INFORMATICS

THESIS APPROVAL SHEET-I

This is to certify that the thesis entitled "**Automatic Fish Species Identification Using Deep Learning Technique**" submitted in partial fulfillment of the requirements for the degree of Master's with specialization in Computer Science, the Graduate Program of the School of Informatics, and has been carried out by Habtamua Zerihun Tsegaye. Therefore we recommend that the student has fulfilled the requirements and hence hereby can submit the thesis to the department.

Name of major advisor: Degif Teka (PhD)

Signature: _____

Date of submission: 06/12/2023

Name of co-advisor : Wogene Alemayehu (MSc)

Signature: _____

Date of submission: Dec 6,2023

THESIS APPROVAL SHEET-II

We, the undersigned, members of the Board of Examiners of the final open defense by Habtamua Zerihun have read and evaluated her thesis entitled "**Automatic Fish Species Identification Using Deep Learning Technique**", and examined the candidate. This is, therefore, to certify that the thesis has been accepted in partial fulfillment of the requirements for the degree of Master of Science in "Computer Science".

Name of Major Advisor	Signature	Date
Degif Teka (PhD)		06/12/2023
Name of Internal Examiner-I	Signature	Date
Efrem Yohannis (PhD)		29/11/2023
Name of Internal Examiner-II	Signature	Date
Tegegn Gobena		29/11/2023
Name of External examiner	Signature	Date
Asrat Mulatu (PhD)		29/11/2023
SGS Approval Ephrem Alamerew Engida Dean, Faculty of Informatics	Signature	Date
		29/11/2023

ACKNOWLEDGMENTS

First and foremost, I express my gratitude to the divine entity, God, who is the ultimate source of wisdom and the origin of knowledge. I offer my praise to Him for His invaluable assistance throughout the completion of this work. Additionally, I extend my gratitude to the Ever-Virgin, St. Mary, the mother of our Lord.

I would like to convey my deepest appreciation to my advisor, Dr. Degif Teka. I am truly thankful to him for sharing his insights and providing constructive feedback. Engaging in discussions with him has constantly reinforced my belief in the realm of possibilities. I aspire to become an advisor as exceptional as him one day, guiding my own students just as he has guided me.

I would also like to express my sincere thanks to my co-advisor, Mr. Wogene Alemayehu for giving me his endless support, motivation and valuable comments.

Additionally, I would like to acknowledge the domain experts at the Center for Aquaculture Research and Education (CARE) who are currently employed there. Their expertise and efforts in certifying the labels of each species of fish have been instrumental in the success of this research.

Finally, I want to express my gratitude to my mother Lemlem Mande, my sister Netsanet Zerihun and my father H/Michael G/Tsadiqe for providing me with unfailing support and continuous encouragement throughout my years of study. Thank you everyone for being with me. Thank you.

STATEMENT OF THE AUTHOR

I hereby declare that this MSc thesis is my original work and has not been presented for a degree in any other university, and all sources of material used for this thesis have been duly acknowledged.

Name of the student: Habtamua Zerihun Tsegaye

Signature: 

Date of Submission: DEC 06, 2023

Place: Institute of Technology, Hawassa University, Hawassa

ABSTRACT

In recent years, the growing global population has led to an increased demand for animal protein, including fish and other aquatic products. Aquaculture has emerged as a primary method for meeting this demand. There is a need for reliable and accurate methods to identify fish species. However, the accurate identification of fish species remains a challenge as there are various fish species endemic to different regions. This research focuses on addressing this challenge by developing a system for automatic fish species identification using deep learning technique, with a specific emphasis on convolutional neural network (CNN).

To accomplish the objective of the research, fish species images were collected from Lake Hawassa. The collected dataset was certified by domain experts from the Centre for Aquaculture Research and Education (CARE) at Hawassa University. A custom dataset was prepared, consisting of a total of 6000 images of six fish species: Oreochromis niloticus, Clarias garipienus, LabeoBarbus intermedius, Barbus paludinosis, Garra quadrimaculata, and Aplocheilichthys. The proposed system for fish species identification implements a preprocessing module that involves image resizing and pixel value normalization to ensure uniformity and enhance training performance. Data augmentation techniques were utilized to generate diverse training examples. For classification, convolutional neural network (CNN) is employed, either trained using Convolutional neural network (CNN) architectures or utilizing pre-trained models such as Inceptionv3, VGG16, and ResNet50. Evaluation metrics were employed with two different dataset ratios: 70/30 and 80/20 and also three pre-trained models were used for comparison. The results demonstrate that our proposed model 70/30 ratio outperforms the pre-trained models in terms of training, testing accuracy, as well as loss. Our model achieved a training accuracy of 100%, validation accuracy of 99.7% and a testing accuracy of 99.5%, indicating better learning and classification capabilities. Additionally, the model achieved a recall, precision and f1 score of 100%. This research contributes to the field of fish species identification. By leveraging deep learning techniques, Particularly CNN, our model achieves better accuracy in automatic fish species identification. It reduces reliance on expert skills, addresses unresolved problems, and contributes to the progress of accurate fish species identification.

Keywords: Fish species identification, deep learning, CNN, aquaculture, endemic fish species.

Table of Contents

List of Tables

List of Figures

List of Algorithms

Acronyms/Abbreviations

ANN	Artificial Neural Networks
CARE	Center for Aquaculture Research and Education
CNN	Convolutional Neural Network
DL	Deep Learning
DNN	Deep Neural Network
DBN	Deep Belief Network
DBM	Deep Boltzmann Machine
RBM	Restricted Boltzmann Machine
RNN	Recurrent Neural Network
ReLU	Rectified Linear Unit
FD_Net	Fish Detection Network
RGB	Red-Green-Blue
ROI	Region of Interest
SVM	Support Vector Machine
GPU	Graphics Processing Unit
COCO	Common Objects in Context
VGG	Visual Geometry Group
ILSVRC	Large Scale Visual Recognition Challenge
FFANNS	Feed Forward Artificial Neural Networks
ResNet	Residual Network

CHAPTER 1: INTRODUCTION

1.1. Background

Nowadays, the world population's growth has resulted in an increasing need for animal protein. Fish and other aquatic animal products are essential for improving nutrition, providing vital fatty acids, protein, and micronutrients [1]. To meet this demand, aquaculture, the aquatic counterpart of agriculture, has rapidly emerged as the primary method for increasing food production from aquatic ecosystems [2]. Researchers in the field of aquaculture have recently shown a keen interest in harnessing machine learning technologies, particularly deep learning, for various tasks such as fish biomass assessment, age-based fish classification, prediction of water quality, behavioral analysis, and fish species identification [3]. These technologies have the potential to enhance efficiency and sustainability in aquaculture practices.

Fish species exhibit remarkable diversity globally, with more than 33,000 species differing in terms of their shape, lifestyle, size, color, quality, and price [4]. However, many of these species have similar visual traits, making it challenging to differentiate them based on their external appearance. And Ethiopia, with its unique geographic conditions, stands out for its rich biodiversity, including various fish species. A notable location is Lake Hawassa in the Sidama Region of Ethiopia's Rift Valley, renowned for its natural beauty and diverse ecosystem. It is a large lake spanning approximately 129 square kilometers with a maximum width of 20 kilometers and a length of 25 kilometers. It has an average depth of 7 meters, but some areas can reach depths of up to 15 meters [5]. The lake's depth can vary based on seasonal rainfall and water levels.

Compared to other lakes in the Ethiopian Rift valley, the lake is exceptionally bio-diverse since it has a large number of species of various kinds of plants, animals, and microorganisms and it is also home to six fish species, those are the endemic African species Nile Tilapia (*Oreochromis niloticus*), the commercially important African catfish (*Clarias gariepinus*), and other species like the African big barb (*LabeoBarbus intermedius*), straight fin barb (*Barbus paludinosus*), the stone lapping minnow (*Garra quadrimaculata*), and the black lampeye (*Aplocheilichthys*). The lake's biodiversity contributes to its value as a source of income from ecotourism and provides recreational opportunities such as bird-watching, fishing, and boating for both locals and foreign visitors.

In this context, a study is underway to develop an automatic fish species identification system using deep learning techniques, particularly convolutional neural networks (CNNs). Features

that were manually designed are prone to human error. These days, Deep learning, a subfield of machine learning, has gained attraction in various fields due to its ability to learn hierarchical representations of data [6]. CNNs, a type of deep learning model, have demonstrated excellent performance in image recognition and classification tasks. Deep learning has a wide range of applications, including object detection, object recognition/identification, image classification, and natural language processing. Object detection and object recognition are similar techniques for recognizing objects; they differ in how they are carried out. Object recognition helps identify items in pictures and videos. While, finding instances of objects in an image is object detection. Using deep learning, object detection is a subset of object recognition, which involves not only identifying the object in the image but also locating its location.

The study aims to train a CNN model using a dataset of recorded fish images, enabling the model to learn the distinguishing features and patterns associated with different fish species. The successful development of this automatic identification system can have significant implications for scientific research, fisheries management, aquatic ecotourism, and the advancement of sustainable aquaculture practices.

1.2. Motivation

The motivation behind this study stems from the widespread use of advanced machine learning techniques in aquaculture for tasks such as fish biomass evaluation, fish classification, behavioral analysis, and water quality forecasting. As aquaculture plays a crucial role in providing high-quality protein worldwide and is a rapidly growing sector in the food production industry [7], there is a need for reliable and accurate methods to identify fish species.

Manual identification of fish species based on morphology is prone to inaccuracy, time-consuming, and challenging due to the large number of species and their close resemblance to one another [8]. This highlights the importance of developing an automated fish identification system that can provide efficient and precise results.

In Ethiopia, there is a unique opportunity for fish species identification due to the country's rich biodiversity, particularly in its freshwater ecosystems. Ethiopia is home to numerous endemic fish species that have evolved distinct characteristics in isolation, making them of great interest to researchers and conservationists. The Rift Valley lakes in Ethiopia are renowned for their stunning beauty and provide diverse habitats for a wide range of fish

species, contributing to high species richness and endemism, making them important areas for fish species Identification and conservation efforts.

Furthermore, fish species in Ethiopia have adapted to specific environmental conditions, such as high altitudes, varying water temperatures, and different water chemistry. These adaptations have led to the evolution of distinct morphological and behavioral traits, making fish species identification in Ethiopia particularly interesting and challenging.

Although Ethiopia has made significant efforts in studying and conserving its fish species, the exploration of machine learning technologies for fish species identification is relatively limited. By leveraging deep learning techniques, particularly in the field of fish species identification, it is possible to improve identification accuracy and efficiency, benefiting commercial activities and supporting conservation efforts.

1.3. Statement of the problem

The existing research on fish species identification has been conducted in various countries using different approaches. However, to the best of the researcher's knowledge, in Ethiopia, there is no research work carried out for fish species identification.

Ethiopia is home to a diverse range of freshwater fish species. Based on the recent findings from different inland water bodies of the country, there are 200 fish species in different drainage basins [9]. According to this study, 191 fish species are indigenous (native) and 9 exotic fish species. From those 191 fish species 53 species are endemic to Ethiopia. All the six fish species that are found in Lake Hawassa are native to Ethiopia but not endemic.

The identification mechanisms developed for other countries may not accurately identify the endemic or native species found in Ethiopia. Additionally, similar fish species found in different countries and found in the same country in different places differ from each other due to several factors such as water temperature or salinity, Environmental adaptation (water quality, food availability) and geographical features (mountains, rivers, lakes, or oceans).

Traditionally, fish species identification relies on morphological keys and comparison of sample characteristics with taxonomic keys found in literature and preserved specimens [10]. However, this method can be time-consuming and challenging, especially when dealing with species that have similar characteristics, particularly during the larval stage [11].

To address these challenges and improve on existing methods, the proposed solution is to develop a deep-learning identification mechanism for fish species in Ethiopia. The research aims to answer the following questions:

1. How can a model be developed that is capable of accurately identifying fish species?
2. What are the optimal hyperparameters to be used in the convolutional neural network (CNN) model to achieve the best performance?
3. To what extent is the proposed convolutional neural network (CNN) model effective for the identification of fish species in Ethiopia?

1.4. Objectives of the study

1.4.1. General Objective

The main objective of this research work is to develop a model for identification of fish species using deep learning technique.

1.4.2. Specific Objectives

To meet the general objective of this research work, the following specific objectives are identified.

- To review related literature to help understand the features and approaches used in fish species classification.
- To collect and prepare a suitable dataset (image dataset) from Hawassa lake for training and testing purposes.
- To combine different convolutional neural network (CNN) hyperparameters to increase the performance of the fish species identification model.
- To design a model for fish species identification.
- To evaluate the performance of the proposed model with evaluation metrics and pre-trained models.

1.5. Scope and Limitation of the study

The scope of this study is concerned with the development of automatic identification of fish species. In Ethiopia, there are more than 200 fish species[12], but we have considered only the 6 fish species that are available in Hawassa lake (*Oreochromis niloticus, Clarias garipienus, LabeoBarbus intermedius, Barbus paludinosis, Garra quadrimaculata* and *Aplocheilichthys*).

The first four are known to be mostly available while the rest are not easily available.

This study mainly focuses on identifying fish species into the corresponding class. Recommending the appropriate fish species for food, teaching purpose, different medical purpose, and weather conditions in the aquaculture sector is beyond the scope of the study in our work. In this thesis, we have concerned only with fish species identification from Hawassa Lake, and the fish species in other lakes in Ethiopia are not considered as it is time bound and resource matters.

1.6. Methodology

In order to achieve the objective of this research, we have used a design science research methodological approach. Design science research methodology creates and evaluates information technology artifacts that can solve organizational problems [13]. Design science research methodology has a certain phase such as (problem identification, data collection, design, evaluation and summary) to develop the artifact. The following sections describe the methodology steps of this study. In this section, different methods of literature review, data gathering, design and development tools, and evaluation techniques are discussed. These techniques are used to identify the concepts that facilitate solutions to the fish species identification model.

1.6.1. Literature Review

Different literatures that are important to this research work are reviewed, including books, articles, research papers, and materials related to the subject that assist us in choosing efficient algorithms. This is done in order to define the problem and understand the state of the art in the area of designing automatic fish species identification models.

1.6.2. Data Collection

In order to accomplish the objective of the research the samples of fish species images were collected from Hawassa Lake. All the species of the sample are certified by the domain experts of the Centre for Aquaculture Research and Education (CARE) Hawassa University. The image was taken with a smartphone's camera.

1.6.3. Design and Development Tools

The implementation of the proposed model requires some image pre-processing, image analysis, identification tools and application development environment studies. In this phase, image processing and identification were determined. To design our CNN model, we used Keras (with TensorFlow as a backend). To implement the source code, we have used the

Python programming language. The design of the model is based on deep learning technique. To implement the proposed model for the identification of fish species, one of CNN's deep learning technique is used. The identification systems were supervised because classes were predefined that correspond to the selected species by domain experts during dataset collection.

1.6.4. Evaluation technique

Once the solution reaches a sufficient state, the design artifact can now be evaluated. A solution to a problem was evaluated by using the test dataset on the classifier that was built by the training dataset in order to test the designed artifact or to measure how effective the developed model was. The performance of the model was evaluated by comparing its output with the observed data using performance measuring metrics and percentage accuracy measures for each class such as recall, f1-score, and precision values for evaluating accuracy. A confusion matrix is used to determine if samples were correctly or incorrectly assigned to their respective class, driving true positives, false positives, true negatives, and false negatives[14].

1.7. Significance of the study

Some of the significance of this study has listed below:

- For many people, it may be challenging to identify different fish species. Because aquatic life is rarely observed and is mostly unknown to most of us. It is crucial to have a system in place that can identify different species of fish. It will be very useful for those who want to study different types of fish species.
- In Ethiopia there are many fish species such knowledge of knowing fish species will be useful to tourists in the context of promoting aquatic ecotourism.
- The dataset for this study was created entirely from scratch. It enables researchers who need to take part in achieving the goal of developing efficient deep learning techniques for further investigation.
- The end result of this research work helps identify fish species more accurately, particularly in our country. It also serves as a starting point for future researchers who need to concentrate on species identification in the agricultural and aquaculture sectors.
- Centre for Aquaculture Research and Education (CARE) experts also benefit from the research in that it will be used as a decision support tool and to train students.

1.8. Organization of the Thesis

This section presents an overview on the contents of all chapters. This research work is organized into five different chapters.

The **first chapter** presents a preliminary introduction to this study. It provides the general structure included in this study. It thus provides enough background information to help the reader understand the reason behind the study and what the researcher plans to accomplish by carrying out the research. The chapter provides an overview of the whole study.

In **Chapter Two**, literature is reviewed on the concept of different research work using machine learning and deep learning is presented. In addition, we have given a detailed description of the building blocks of CNNs and its architectures which have been used in the field of image processing and also we describe some pretrained models with its architectures.

A detailed analysis of different works related to the fish species identification is also presented. Only those works that are related to our work are discussed.

In **Chapter Three**, a detailed description of the proposed system is discussed. The components that compose the system (dataset preparation, preprocessing, Data Augmentation, training, and classification) using deep learning techniques and the responsibility of each component are described in detail.

In **chapter four,** experimental evaluation of the proposed model for automatic identification of fish species is described in detail. The proposed model's implementation and the dataset used are both fully detailed. Finally, the test results are compared with the state-of-the-art models and related works.

In **Chapter five** conclusions, future work, and the contributions of this research work are presented.

CHAPTER 2: LITERATURE REVIEW AND RELATED WORKS

2.1. Introduction

Reviews of literature related to an overview of commercially important fish species in Ethiopia, the steps of image processing techniques, different research works using machine learning and deep learning, a detailed analysis of different related research works are presented, and also summarized an overview of related work by other scholars or authors that is relevant to the study.

2.2. Overview

Fish contributes to global food security and has historically played a significant role in many countries, contributing 15-20% of the daily requirement for animal protein[15]. There are numerous lakes and rivers in Ethiopia that are used for fish production, and various fish species found in these lakes and rivers. Because fish is so nutrient-dense, even small portions can help individuals eat healthier food. Fish may significantly improve Africa's food security and nutritional condition especially as a source of food for low income, more than 200 million Africans consume fish on a daily basis [16].

2.3. Fish species

Fishes are incredibly diverse and can be divided into numerous groups. For instance, fishes are diverse with respect to species number, body sizes, habitat type, sexuality, breeding behavior, feeding behavior, age, locomotion, toxicity, vulnerability, etc[17].

Commercially important fish species are those that are harvested for human consumption, trade, or livelihood. Examples of most common or notable commercially important fish species in Ethiopia are Nile tilapia (*Oreochromis niloticus*) is widely distributed and cultivated in Ethiopia [17]. It is an important source of food and income for many people, Nile perch (*Lates niloticus*): This is a large predatory fish that is native to the Nile basin, including Lake Tana in Ethiopia, and Ethiopian straightfin barb (*Barbus pleurogramma*): This is a small barb that is endemic to the Awash basin and several rift lakes in Ethiopia.

The study was conducted on six fish species from Lake Hawassa. Around Lake Hawassa Nile tilapia (*Oreochromis niloticus*), African catfish (*Clarias gariepinus*), and Barbus (golden) fish (*Labeobarbus intermedius*) are three species with significant commercial importance [18]. About 90% of the total production is made up of Nile tilapia, while the contributions from Barbus (golden) fish and African catfish are only 8% and 3%, respectively, of the total

production. Commercially important fish species reported in Ethiopia are listed in table 2.1 [17].

Table 2.1: Commercially important fish species in Ethiopia

No	Scientific name	Common name	Local name
1	*Lates niloticus*	Nile perch	Nech asa
2	*Oreochromis niloticus*	Nile tilapia	Qoroso/Chogofe
3	*Barbus species*	Barbus	Bilicha
4	*Labeo species*	Labeo	Barbo/Lebi
5	*Clarias garipienus*	Catfish	Ambza
6	*Bagrus dockmac*	Bagrus	Kerkero
7	*Polypterus bichir*	Nile bichir	Eguwella
8	*Gymnarchus niloticus*	Gymnarchus	Wit
9	*Malapterurus species*	Malapterurus	
10	*Crussian carp*	Carp	Daba
11	*Distichodus niloticus*	Distichodus	Piro
12	*Hydrocynus forskali*	Hydrocynus	Weri
13	*Hetroticus niloticus*	Hetroticus	Ediwela
14	*Citharinus citharinus*	Citharinus	Ajaka
15	*Synodontis species*	Synodontis	Akok

Digital Image processing and computer vision are two related fields that use computers to analyze and understand digital images. Computer vision aims to extract high-level information from images, such as objects, faces, scenes, or actions. Image processing focuses on manipulating images, such as enhancing, filtering, transforming, or compressing them. Currently, both image processing and computer vision have many real-world applications in various domains, such as Agriculture, Health, security and entertainment. Below we present a detail discussion for digital image processing.

2.4. Digital image processing

The term digital image refers to the digital representation of a physical object in a computer. A digital image is made up of picture elements called pixels, each of which has a finite, discrete numerical representation of the gray level or intensity value produced by a two-dimensional function called f(x, y), where x and y are spatial (plane) coordinates on the x and y-axis. An approximate representation of a real scene that was acquired using any digital

image capturing device is used throughout the digitization process. An image is taken to be rectangular in shape with x rows and y columns [19].

A technique for processing images using various computer algorithms is called digital image processing. It manipulates digital images considering various parameters such as color processing, image recognition, video processing, etc. In this method, a digital image serves as the input, and the output could either be the image itself or some of its features. A system that can distinguish between a specific object and another object based on the characteristics of an image itself can be created. An image can be manipulated for a variety of reasons, including improving the visual information for human perception, storage, transmission purposes, or autonomous machine perception.

Three stages can be visible in digital image processing. These levels are termed as low, intermediate level or middle-level and high level processes. Primitive operations like noise reduction in image preprocessing, contrast enhancing, and image sharpening are examples of low level processes. When both inputs and outputs are images, a process is said to be low level. Mid-level processing involves tasks such as segmentation, feature extraction. Unlike low level processing, in mid-level its inputs are images, but its outputs are attributes extracted from those images. High level processing involves making sense of a group of recognized objects like classification, tracking etc.

The processing of digital images by a digital computer is referred to as digital image processing. Basically, different image processing applications can go through different stages. However, the basic steps that every image processing application passes through are shown in Figure 2.1.

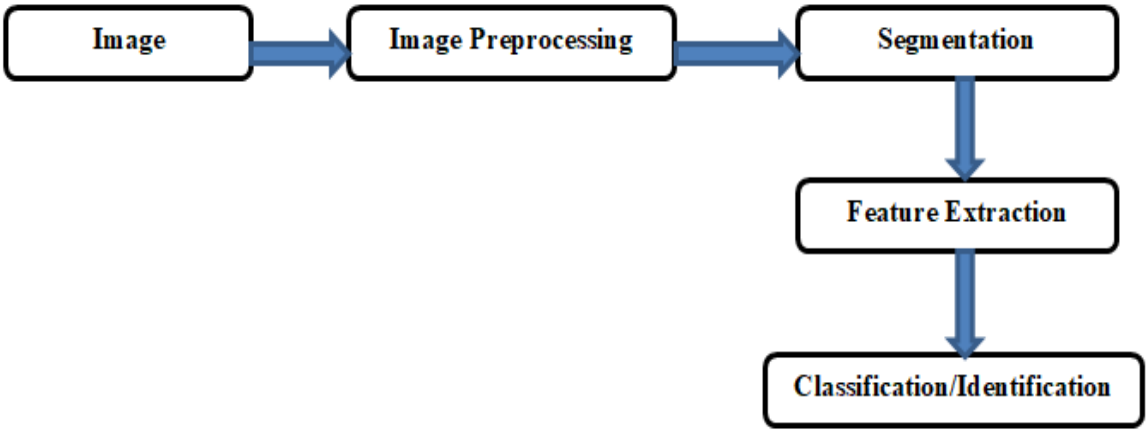

Figure 2.1: Steps of digital image processing

2.4.1. Image acquisition

Image acquisition is the first step of image processing that is done using a digital camera with RGB color and it is loaded into the computer [20]. Fish image acquisition was the beginning step that we considered in fish species identification because without the fish images the other processes would not continue. Having a source input image that operates within a controlled and measured guideline is the aim of image acquisition. Since the image collected during image acquisition is a raw image or unprocessed image, it requires additional image processing techniques.

2.4.2. Image Preprocessing

The pre-processing of acquired images is performed by transforming them for better classification of images. The transformation techniques which are used for pre-processing of the image can be resizing the image, reducing blurring, removal of noise, contrast correcting in the image, watermark removal, etc. Preprocessing, a subfield of image processing, uses smoothing or blurring to the image to reduce noise. Operations on this preprocessing step are done to improve the image by suppressing unnecessary distortions or enhancing major image features important for further identification processes [21]. During the image acquisition noise can come from a variety of sources. This noise happens when the camera either fails to focus the image properly or when there are unwanted environmental signals present. This noise changes the size and shape of the object of an image and blurs the edge information [22]. Image preprocessing is low-level processing to enhance the images before computational processing. Image preprocessing is a technique to bring out detail that is obscured, or simply to highlight certain features of interest in an image.

2.4.3. Image Segmentation

The process of image segmentation is used to identify objects and boundaries (such as lines or curves) in images. The result of image segmentation is a collection of segments that collectively cover the full image or a collection of contours extracted from the image. The segmented objects have often termed the foreground and the rest of the image is the background. It is used to crop out undesired portions of images and locate the region of interest. There are many segmentation techniques, but they can be categorized into detection of discontinuities (boundary approach) and detection of similarities (region approach). The segmentation technique based on detection of discontinuities approach assumes that the boundaries of the regions are different from each other, and the background. Therefore, we

can detect the boundary by considering the local intensity discontinuities. Segmentation techniques such as edge detection, point detection, line detection follow this approach. Contrarily, the recognition of similarities relies on continuities. These techniques divide the entire image into sub regions depending on some similarity rules. Examples of such algorithms include thresholding, region growing, region spreading, region merging etc. A binary image is the result of any segmentation process, regardless of whether discontinuity or similarity is utilized as the segmentation technique [23].

2.4.4. Feature extraction

The primary step in digital image processing after segmentation is feature extraction. Features are information that is taken from images in the form of numerical values that are difficult for humans to understand and correlate. If we consider the image as data the information extracted from the data is known as features. The dimensions of features extracted from an image are substantially less than the dimensions of the original image. The dimensionality reduction reduces the overhead of processing a bunch of images. A good feature set includes discriminating information that can separate one object from another object. The selected set of features should be a small set whose values efficiently discriminate among patterns of different classes while being identical for patterns belonging to the same class. Several image features represent an image for classification/identification systems. Most popular among them are the shape, texture, and color of an image.

2.4.5. Classification

The process of identifying what an image represents (object) based on information available about these objects (measured values of features). Models or classifiers are used to predict the classes of these objects. Targets, labels or categories are all common names for classes. Image classification methods are supervised and unsupervised. In supervised classification classes of an object may be specified prior by an analyst whereas in unsupervised learning objects are automatically clustered into sets of classes.

There are two steps in classification: namely, the learning step and the classification step. In the learning phase, the model or the classifier is built by a set of the training data set and their associated class labels whereas, in the classification phase, the classifier utilizes specified features of an object as input to classify a new image (different from the training set) into one of the class labels of an object.

2.5. Approaches to classification or pattern recognition algorithms

There are several approaches to classification or pattern recognition algorithms. Here are some commonly used approaches:

2.5.1. Support Vector Machine

Support Vector Machines (SVM) is a popular classification algorithm used in machine learning. It is capable of solving both linear and non-linear classification problems. The goal of SVM is to find an optimal hyperplane in the feature space that maximally separates different classes of data points. The hyperplane is chosen in such a way that it maximizes the margin, which is the distance between the hyperplane and the nearest data points from each class. By maximizing the margin, SVM aims to achieve better generalization and improve the model's ability to classify new, unseen data accurately [24].

SVMs can handle linear classification problems where the data points are linearly separable. However, they can also handle non-linear classification problems by using techniques such as the kernel trick. The kernel trick allows SVMs to transform the data into a higher-dimensional feature space, where it becomes linearly separable. Common kernel functions used in SVMs include linear, polynomial, Gaussian radial basis function (RBF), and sigmoid kernels. SVMs have several advantages, such as their ability to handle high-dimensional data, good generalization performance, and robustness to overfitting. However, SVMs can be computationally intensive, especially when dealing with large datasets.

Here is an example of previous literature conducted by researchers using SVM or support vector machine.

Ethiopian maize diseases recognition and classification using support vector machine

The study entitled "Ethiopian maize diseases recognition and classification using support vector machine" by Enquhone Alehegn [25], addresses the challenge of identifying and classifying maize leaf diseases in Ethiopia. The author highlights that there are more than 72 known maize diseases in Ethiopia, which affect different parts of maize plants. Traditional methods of disease identification and classification, such as chemical analysis or visual observation, have limitations in terms of time consumption and the need for professional expertise. Therefore, the study aims to develop an automated approach using image processing and support vector machine (SVM) modeling. The researcher collected a dataset of 800 maize leaf images. The dataset was split into training and testing sets, with 80% used for training the SVM model and the remaining 20% for testing. The study employed

combined texture, color, and morphology features extracted from the images, which were then used as inputs to the SVM model. The experimental results showed an average accuracy of 95.63% achieved for maize leaf disease recognition and classification. This indicates that the developed approach using SVM and image processing techniques is effective in accurately identifying and classifying maize diseases based on the collected dataset.

This study contributes to the field of automated disease recognition and classification in agriculture, specifically focusing on maize diseases in the Ethiopian context.

2.5.2. Naive Bayes algorithm

One of the commonly used approaches for Classification or pattern recognition algorithms is the Naive Bayes algorithm. Naive Bayes is a probabilistic algorithm that assumes the features are conditionally independent given the class label. It calculates the probability of a sample belonging to each class based on the observed feature values, using Bayes' theorem. By comparing the probabilities, Naive Bayes assigns the sample to the class with the highest probability [26]. While the assumption of feature independence may not hold in real-world scenarios, Naive Bayes remains popular and effective, particularly in text classification tasks. It is computationally efficient, robust to irrelevant features, and has been widely employed in spam filtering, sentiment analysis, and document categorization. Despite oversimplifying the relationships between features, Naive Bayes can still provide accurate results when the independence assumption is reasonably met or with sufficient training data.

This is an example of prior research done by a researcher who used the Naive Bayesian Classifier.

Diagnosis of Alzheimer's disease using Naive Bayesian Classifier

The research paper [27] on *"Diagnosis of Alzheimer's Disease using Naive Bayesian Classifier"* by Bhagya and Sheshadri addresses the urgent need for early detection of Alzheimer's disease (AD) and its impact on individuals worldwide. By conducting a study with 466 subjects using neuropsychological tests, the authors focus on utilizing the Naive Bayesian classifier, a probabilistic algorithm, to accurately diagnose AD based on test results. The paper highlights the significance of early diagnosis and the prevalence of AD, emphasizing the potential to enhance the quality of life for affected individuals. The authors also introduce the CSID battery, specifically designed for primary AD diagnosis, and discuss the advantages it offers over other existing batteries. By evaluating the proposed model using WEKA and the Naive Bayesian classifier, the study provides insights into the application of

machine learning techniques for AD diagnosis. Overall, this research contributes to the understanding and potential improvement of AD diagnosis, furthering the goal of early intervention and improved patient outcomes.

2.5.3. Artificial Neural Networks (ANN)

The way biological nerve systems process information serves as the basis for the information processing paradigm known as Artificial Neural Networks (ANNs) [28]. ANNs are computational models that aim to mimic the structure and function of biological neural networks found in the human brain. It is consists of interconnected nodes, or neurons, organized into layers. The three main types of layers in an ANN are the input layer, hidden layers, and output layer.

The input layer serves as the entry point for data into the network. The number of neurons in the input layer typically corresponds to the number of input features provided to the network. Each neuron in the input layer receives a specific input value or feature.

Hidden layers are one or more layers located between the input and output layers. These layers play a crucial role in the network's ability to model complex relationships and extract meaningful patterns from the input data. The number of neurons in each hidden layer can vary, and it is a design choice that depends on the complexity of the problem being addressed.

The output layer provides the final results or predictions based on the information processed by the previous layers. The number of neurons in the output layer depends on the nature of the problem being solved. For regression tasks, where the goal is to predict a continuous value, the output layer typically has a single neuron. In classification tasks, where the goal is to assign input data to predefined classes, the number of neurons in the output layer corresponds to the number of distinct classes.

The connections between neurons are represented by weights, which determine the strength of the connection between two neurons. In a fully connected feed-forward network, each neuron in a particular layer is connected to every neuron in the adjacent layer. These weights are initially assigned random values and are iteratively adjusted during the training phase to optimize the network's performance. During training, the network learns from a labeled dataset, adjusting the weights to minimize the difference between the predicted outputs and the true labels. This optimization process is typically done using an algorithm called back propagation, which calculates the gradient of the network's error and updates the weights accordingly.

Once the network is trained, it can be used in the testing phase with new, unseen data. The input data is fed through the network, and the output layer produces predictions or classifications based on the learned patterns and relationships discovered during training.

The behavior and performance of an ANN are influenced by its architecture, including the number of layers, the number of neurons in each layer, and the types of connections between layers. The choice of activation functions, which determine the output of each neuron, also plays a significant role in shaping the network's behavior.

ANN can be classified into shallow and deep learning. Shallow learning has less number of neurons compared to Deep learning (DL) [28]. Image classification has been done on both learning methods. But there is a fundamental difference among them. Shallow learning uses one neuron for each input which increases the number of parameters. The problem became difficult when the image size increased because the amount of weight became larger.

Another main difference is shallow learning doesn't extract features by itself. It needs a feature extractor. Feature extractors are problem dependent and must be rewritten for each new dataset. The efficiency of the extractor depends on the assumption of the person who designs the extractor because we simply cannot conceive every possible feature that will be useful [29]. Hence, using shallow learning for image classification tasks is not advisable. So we choose deep learning for our research work.

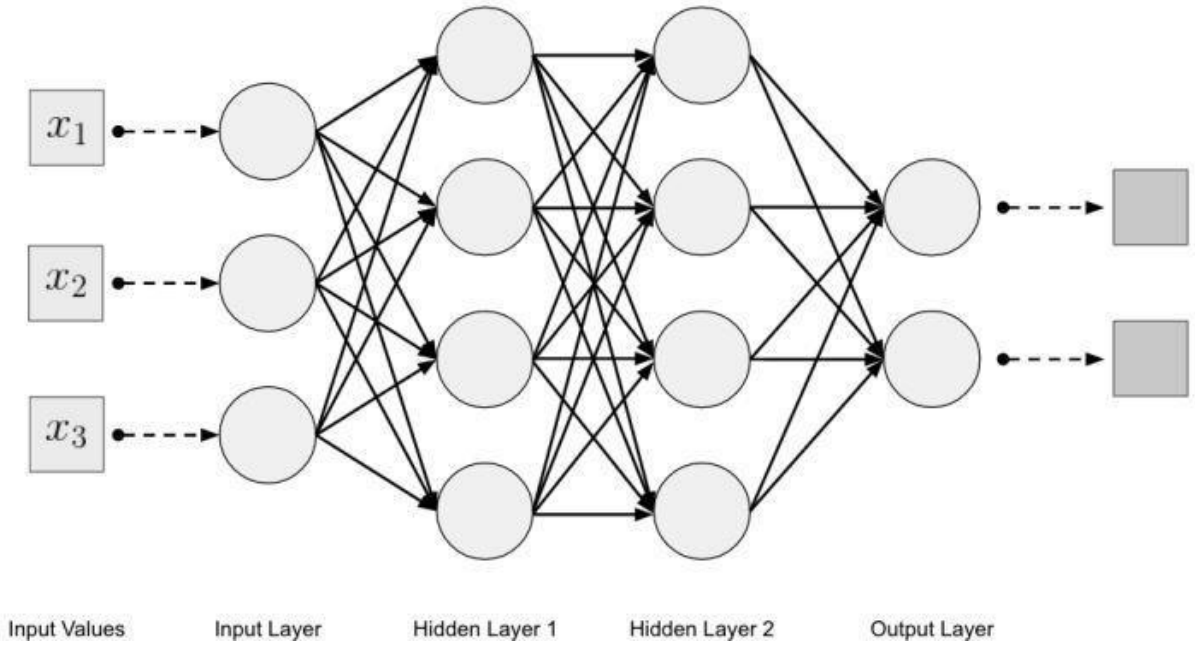

Figure 2.2: Fully Connected Multilayer Feed-Forward Neural Network Architecture

Here is an example of previous literature conducted by researchers using Artificial Neural Network (ANN).

Plant Disease Detection and Classification

The research paper [30] on "Plant Disease Detection and Classification Using Artificial Neural Network" by Endrias presents an approach for automating the detection and classification of plant diseases using artificial neural networks. The study addresses the limitations of manual methods and explores the potential of combining image processing and machine learning techniques. The proposed approach achieved accuracy of 97.6% over a test dataset of 1201 images, outperforming conventional machine learning techniques. The paper also discusses the benefits of automated disease detection for plant health and suggests future work to expand the dataset and enhance the model.

The research work [31] entitled "Detection and Classification of Leaf Disease Using Artificial Neural Network" addresses the challenges faced by farmers in detecting and classifying plant diseases using traditional naked-eye observation methods. The study focuses on developing a simple disease detection system specifically for cotton diseases. By analyzing leaf features through captured images, the proposed system aims to determine the health status of cotton plants. The research employs image processing techniques to extract color features such as HSV from segmented results. An Artificial Neural Network (ANN) is then trained using these feature values to distinguish between healthy and diseased samples. Experimental results demonstrate that the ANN-based classification achieves an accuracy of 80%. The research work proposes a methodology for early and accurate detection of cotton leaf diseases, utilizing diverse image processing techniques and ANN. The approach presented in this study offers a computationally efficient method for recognizing leaf diseases and can be utilized in agricultural applications for disease detection and classification.

2.5.4. Deep Learning techniques

Artificial neural networks are algorithms used in deep learning that are modeled after the structure and operation of the brain. The performance of this type of model improves by training with more examples and by increasing its depth or representational capacity. Another frequently mentioned advantage of deep learning models, in addition to scalability, is their capacity for autonomous feature extraction from unprocessed data [32][28].

There are many types of deep learning used for a variety of tasks in Artificial intelligence [33] like a deep neural network, deep belief network, recurrent neural networks, and convolutional neural networks.

A Deep Neural Network (DNN) is a neural network having more than two layers. It can model complex non-linear relationships. Both classification and regression use it. Deep neural networks handle data in complicated manners by using advanced mathematical models.

Deep Belief Networks (DBNs) are probabilistic unsupervised deep learning algorithms. It can be defined as a stack of Restricted Boltzmann Machines (RBM), in which each RBM layer communicates with both the previous and subsequent layers. The nodes are connected across layers, but no two nodes of the same layer are linked. That is, there is no intra-layer communication. The idea behind a DBN is that by creating a deep network, which offers the capability to extract a hierarchical representation of the input at various levels of abstraction, it is feasible to stack more RBMs on top of each other.

Deep Boltzmann Machines (DBMs) are unsupervised, probabilistic, generative models with entirely undirected connections between different layers. It contains visible units and multiple layers of hidden units. Like RBM (Restricted Boltzmann Machine), no intra-layer connection exists in DBM. Connections exist only between units of the neighboring layers network of symmetrically connected stochastic binary units. DBM can be organized as a bipartite graph with odd layers on one side and even layers on one side units within the layers are independent of each other but are dependent on neighboring layers. Deep Boltzmann Machine (DBM) has entirely undirected connections.

Recurrent neural networks (RNNs) are a type of artificial neural network that can recognize and predict sequences of data such as text, genomes, handwriting, spoken word, or numerical time series data. The RNNs attempt to address the necessity of understanding data in sequences. Recurrent nets differ from feed-forward nets because they have a feedback loop, which means that the output from step n-1 is fed back to the net to affect the outcome of step n and so on for each subsequent step. This allows recurrent nets to capture temporal or sequential dependencies in the data, such as words in a sentence, notes in a melody, or frames in a video. Recurrent nets can also have a memory of previous states, which can help them learn long-term patterns or context. Recurrent nets are more complex and slower than feed-forward nets, but they are more suitable for tasks such as natural language processing, speech recognition, and video analysis.

2.5.5. Convolutional neural networks

Convolutional neural networks (CNNs) are a deep learning class of DNNs that are commonly applied to the study of computer vision and natural language processing. This is a typical version of the multilayer perceptron found in fully connected networks, and it is an analogy to the arrangement of neural connections in the human brain. One input layer, numerous hidden layers, and one output layer make up CNN specifically. Hidden layers include structurally convolutional layers, activation function layers, pooling layers, fully connected layers, and normalization layers. Compared to other classification algorithms, CNNs require less pre-processing and better results as the number of learning increases.

Our task is classifying the input images, so it is a classification problem and the data we have is labeled and supervised learning. The data type is images that are represented in pixels. One pixel is dependent on the value from its 8 neighbors; the feature is a spatial feature. So we choose one of CNN‘s deep learning techniques for image classification.

The model of a CNN is composed of several building blocks such as convolution layers, pooling layers, and fully connected layers. A typical model consists of repetitions of a stack of several convolution layers and a pooling layer, followed by one or more fully connected layers. Convolution and pooling layers are used to feature extraction, whereas fully connected layers use classification of the extracted features into final output the detail of each layer. Forward propagation refers to the process where input data are turned into output through these stages.

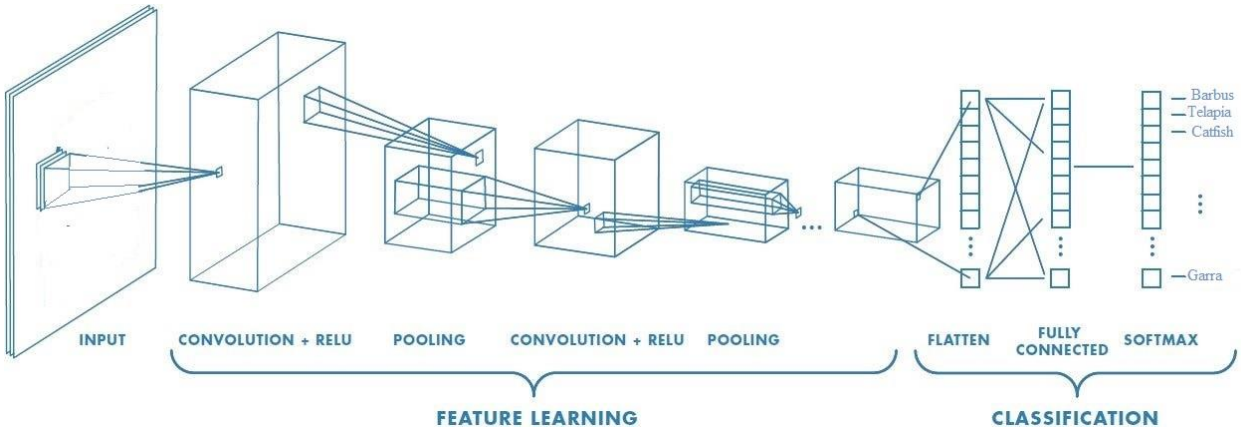

Figure 2.3: Typical CNN Architecture

Convolution layer:

Convolution is a special type of linear operation used for feature extraction, where a small array of numbers, called a kernel, is applied to an input, which is an array of numbers known as a tensor. The output tensor, known as the feature map, is created by repeatedly applying filters (kernel) over the input tensor in element-wise multiplication between each element of the input tensor and the input kernel [34]. This process is repeated using multiple kernels to generate an arbitrary number of feature maps representing different features of the input tensor; thus, different kernels can be considered as extractors for different characteristics. The two major hyper parameters that determine the convolution operation are the size and the number of kernels. Figure 2.4 below is an illustration of convolution operation with a 3×3 filter size. To create the appropriate feature map, we first take a set of 3×3 filters, move them to the right until they touch the end boundary of the input matrix, and then conduct element-wise products (multiplying the image's pixel values by the filter's pixel values). Then, the filter returns to the left, and moves to the right by one pixel. Up until the filter reaches the bottom right corner of the input matrix, this procedure repeats the convolution for every possible pixel location.

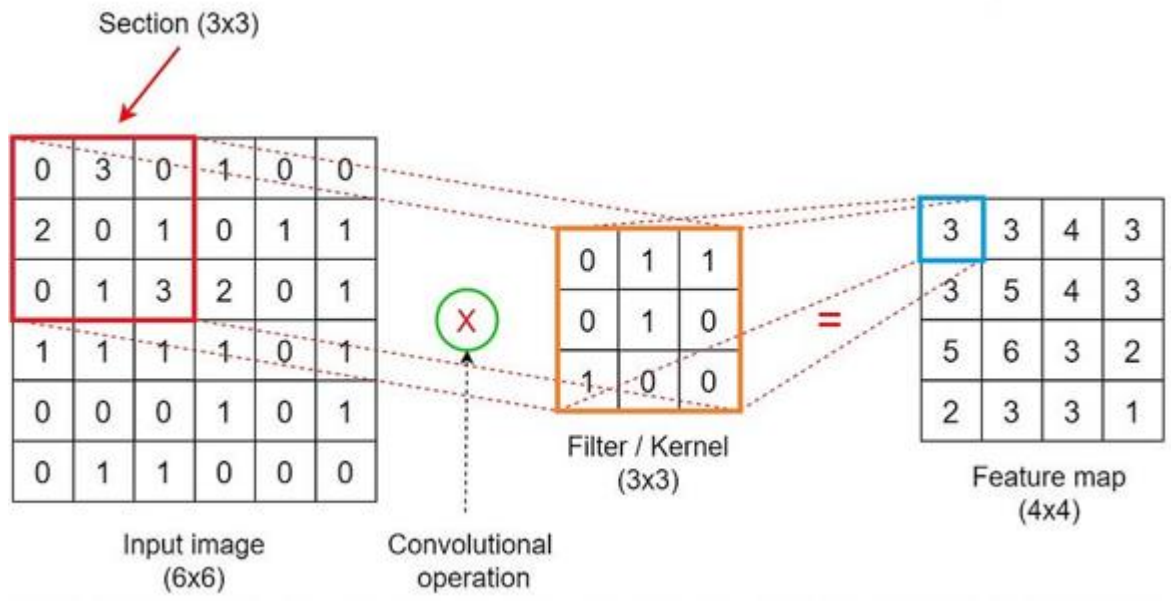

Figure 2.4: Convolution operation with a filter size of 3×3

Last layer Activation Function:

The final fully connected layer typically receives a different activation function than the others. For each activity, a suitable activation function must be chosen. A convolutional neural network may learn complicated patterns in the data by adding an activation function. It is a nonlinear function. It is biologically motivated by how our brains function, where various

neurons fire or are stimulated by different stimuli. In order for multi-layer networks to detect nonlinear characteristics, the activation function must convey nonlinearities to the neural network. As a result, it improves a neural network model's capacity for expression, which enables a deep neural network to fully possess the importance of artificial intelligence by activating neuronal features and resolving non-linear problems.

An activation function is mainly applied to the neural network by taking the output signal from the previous neuron then converting it into some form and deciding what is to be fired to the next neuron of the network [35]. It does some kind of operation on the inputted value and transforms the value between some lower limit and upper limit number. Typical activation functions that are widely used are sigmoid, tanh, and ReLu as discussed in Figure 2.5 below.

Sigmoid

When binary classification is required; a sigmoid function with a range between 0 and 1 is helpful. Every negative value in an input is converted to 0 and every positive number to 1. During training, this activation function may lead to congestion. As a result, it is frequently utilized in binary classification's output layer.

tanh

The range for the hyperbolic tangent function (tanh) is between -1 and 1. Tanh is an updated version of the sigmoid function with a better convergence rate.

ReLU

Rectified linear unit (ReLU) activation functions are frequently employed in CNN and force the output to be zero if the input value is less than or equal to zero. If not, it will set the output value to the same as the input value. The ReLu function's drawback is that if the input value is positive, it does nothing more than forward the value to the next layer node. However, the major advantage considered is that if the ReLU function receives a negative value, it stops the neuron from stimulating by making it 0 and hence preventing the neuron from firing.

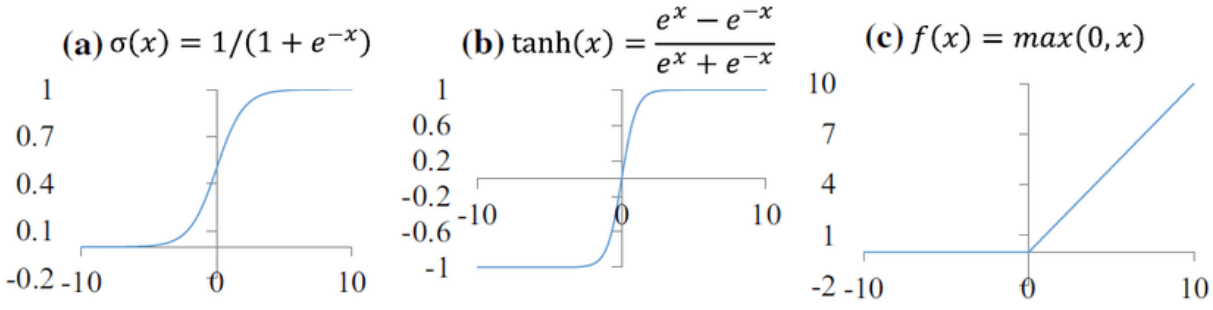

Figure 2.5: Activation functions; sigmoid function, Hyperbolic tangent (tanh) function and Rectifier Linear Unit (ReLu) [36]

Softmax Function

Softmax is used for problems involving a multiclass classification. It produces values in the range of 0 and 1, where the sum of outputs along channels is 1 like not Sigmoid. It is utilized in classification models as the last layer.

Pooling layer:

In order to reduce the number of parameters and the computational complexity of the network, the feature maps' dimensionality is reduced using the pooling layer. The well-known pooling operation in the middle of the CNN architecture is called Max pooling. The "max" function is used to down-sample each activation map in the input tensor in order to minimize the special size. If we extract 2×2 patches from the input tensors using a filter size of 2× 2, no padding, and a stride of 2, the output only retains the largest value and dismisses all other values. The layer can span through the full pixel of the input tensor and reduce it by a factor of 2 when stride and filters of the pooling are both set to 2×2. The max pooling operation is shown in the figure 2.6 below.

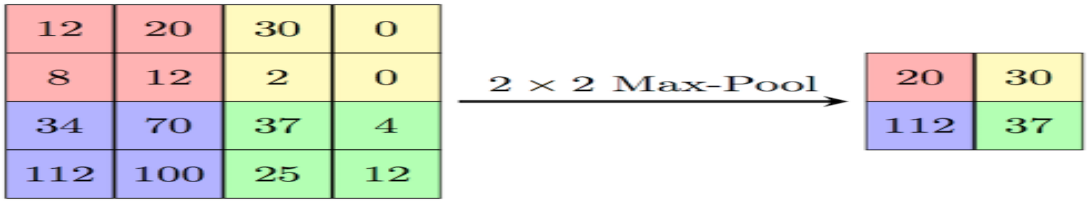

Figure 2.6: Max pooling filter size of 2 × 2

Fully connected layer:

The output of the final convolution layer serves as the input to the fully connected layer, which is flattened (converted into a one-dimensional array). After applying convolution layers and pooling for feature extraction, features are mapped by a subset of fully connected layers to the final outputs of the network before applying the classifier. The number of nodes in the final fully connected layer is the same as the number class.

Dropout layer:

Convolutional neural networks have a huge number of parameters which makes them powerful in machine learning systems. CNN also has a Dropout layer, which is a notable component. The Dropout layer serves as a mask, keeping all other neurons functionality but removing some neurons' contributions to the following layers. Some of the input vector's

features are removed if we apply a Dropout layer; however, some hidden neurons are removed if we apply it to a hidden layer. Dropout layers are essential in the training of CNNs because they prevent the overfitting of the training data [37]. If it is absent, the first set of training samples has an excessively large impact on learning. Consequently, features that emerge only in later samples or batches would not be learned.

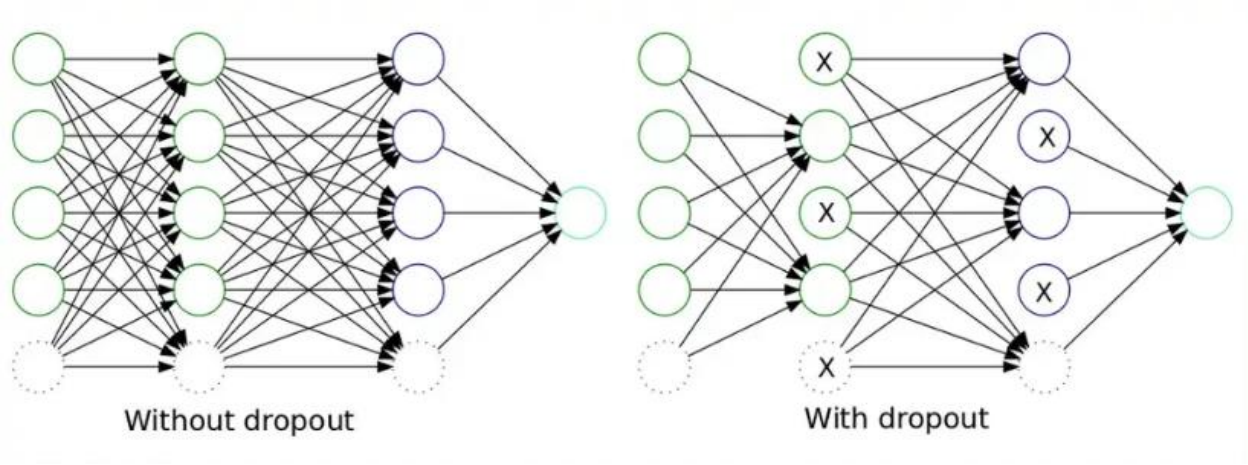

Figure 2.7: Dropout method to prevent overfitting

Numerous research studies have explored various identification mechanisms and classification algorithms across different domains including finance. In the agriculture sector, several researchers have focused on utilizing Convolutional Neural Networks (CNNs) for their work.

Here are some examples of previous literature conducted by different researchers in this field, specifically using CNNs or convolutional neural networks.

Banknote Denomination Classification and Fake Detection System

This research [38] study focuses on developing a deep learning approach for Ethiopian banknote denomination classification and fake detection. The study aims to address the problem of counterfeit banknotes in Ethiopia by using convolutional neural networks (CNNs) for feature extraction and feed-forward artificial neural networks (FFANNs) for banknote classification and fake detection. The study uses real scene images taken from scanners as the input data and applies data augmentation techniques to increase the size and diversity of the dataset. The study achieved accuracy of 99.4% for banknote denomination classification and 96% for fake detection. The study suggests future research using advanced CNN architectures like Google Net and ResNet and larger datasets to improve the performance of the system.

Coffee Leaf Diseases Identification

The research work [39] entitled "Ethiopian Coffee Leaf Diseases Identification Using Deep Learning Features" is by Linigerew and Mogalla presents a deep learning method for detecting four types of coffee leaf diseases that affect Ethiopian coffee production: brown eye spot (BES), coffee berry disease (CBD), coffee leaf rust (CLR), and coffee wilt disease (CWD). The method uses CNNs for feature extraction and SVMs for classification, and achieves high accuracy on real images of diseased leaves. The proposed model achieved an overall classification accuracy of 96.5%. The paper also discusses the importance of controlling coffee diseases and suggests future research directions.

2.6. Transfer learning

Transfer learning is reuse of a pre-trained model as the starting point for a model on a new task. One can attain considerably better performance than training with only a small amount of data by applying transfer learning to a new task. In deep learning it is known as pre-trained models.

There are many benefits we gain from transfer learning such as a much faster and easier fine-tuning of parameters, quick learning of new tasks without defining and training a new network, and suitable for the small amount of training data for the new classification task

In the case of transfer learning, all layers are kept intact except for the last one which is trained for our specific task. This method can be particularly suitable for faster epoch processing times since the layers are frozen and loaded from a previously trained network. It does not also require as much training data as needed for effective classification.

There are different variants of pre-trained neural networks. Each has its own size, architecture, speed, and proportion of benefits and drawbacks. AlexNet, VGGNet, Inception and ResNet are typical examples of models that have the basis of Transfer learning. Below is a discussion of some popular pre-trained CNN architectures including AlexNet, VGGNet, Inception and ResNet.

AlexNet:

One of the most often used CNN architectures is AlexNet. Designed by Alex Krizhevsky, Ilya Sutskever and Geoffrey Hinton. AlexNet won the contest in the ILSVRC-2012 by successfully classifying 1.2 million images into 1000 distinct classes. It achieved a top-5 test error rate of 15.3%, compared to 26.2% achieved by the second-best entry [40]. With a subset

of the ImageNet dataset that contains over 1.2 million training images, 50,000 validation images, and 150,000 testing images, AlexNet is able to handle the image classification problem.

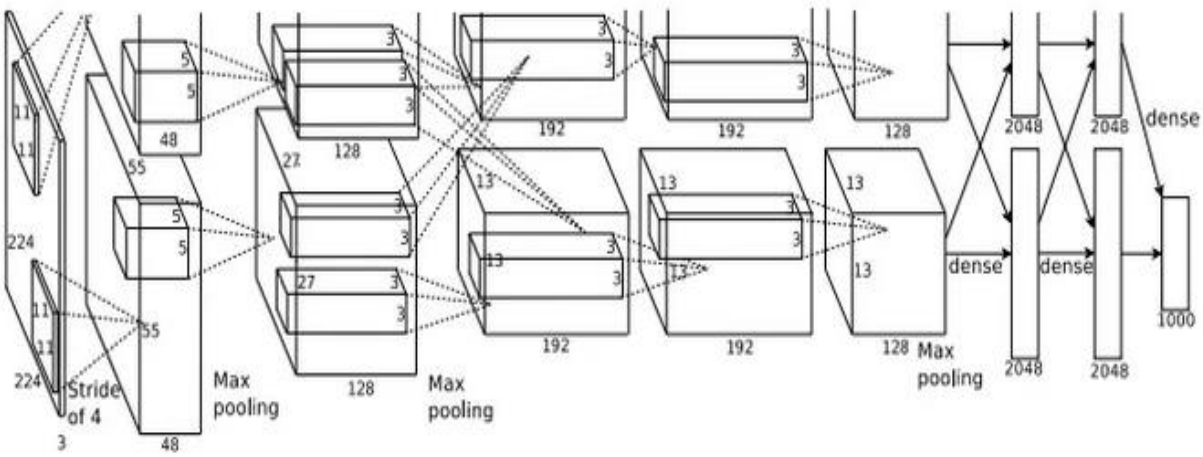

Figure 2.8: AlexNet Architecture

AlexNet has five layers. In the first two convolutional layers, an Overlapping Max Pooling layer comes after each convolutional layer. Convolution layers three, four, and five are all directly related to one another. After the Overlapping Max Pooling Layer, which is connected to fully connected layers, comes the fifth convolutional layer. ReLU is used to increase non-linearity in each dense and convolutional layer to approximate the link between network variables. Each of the fully connected layers has 4096 neurons, and the second fully connected layer feeds data into a classifier with 1000 classifications called a softmax classifier.

The first convolutional layer applies 96 kernels with an 11×11×3 pixel size and a 4 pixel stride to the 224×224×3 input image. The input from the first convolutional layer is convolved in the second layer using 256 filters with a 5×5×48 pixel size. Without any pooling or normalization layers in between, the third, fourth, and fifth convolutional layers are connected to one another. The outputs of the second convolutional layer are connected to 384 kernels of size 3×3×256 in the third convolutional layer. The fourth convolutional layer contains 384 kernels of size 3×3×192, while the fifth convolutional layer contains 256 kernels of size 3×3×192, and 4096 neurons in each of the fully connected layers.

VGGNet:

Another successful CNN network, Visual Geometry Group (VGG), took the second place and gained first place in the localization challenge in the ILSVRC-2014's ImageNet competition. The VGG network architecture was introduced by Simonyan and Zisserman [41]. In order to categorize them into 1000 different classes, it is trained using 1.2 million ImageNet datasets. This network is characterized by its simplicity, using only 3×3 convolutional layers stacked on top of each other in increasing depth. Using max pooling, volume size can be reduced, and then comes a softmax classifier, which is followed by two fully connected layers with a total of 4,096 nodes each.

Training VGG16 and VGG19 was difficult for Simonyan and Zisserman (especially with regard to convergence on the deeper networks), thus to make training simpler, they first trained smaller versions of VGG with fewer weight layers. The main accomplishment of VGG is the significance of deepening the network to increase the effectiveness of visual representations. Compared to popular CNN models from the time, such as AlexNet and ZfNet, the network uses smaller filters and is created with a deeper architecture. VGGNet models like VGG16 and VGG19 are some examples.

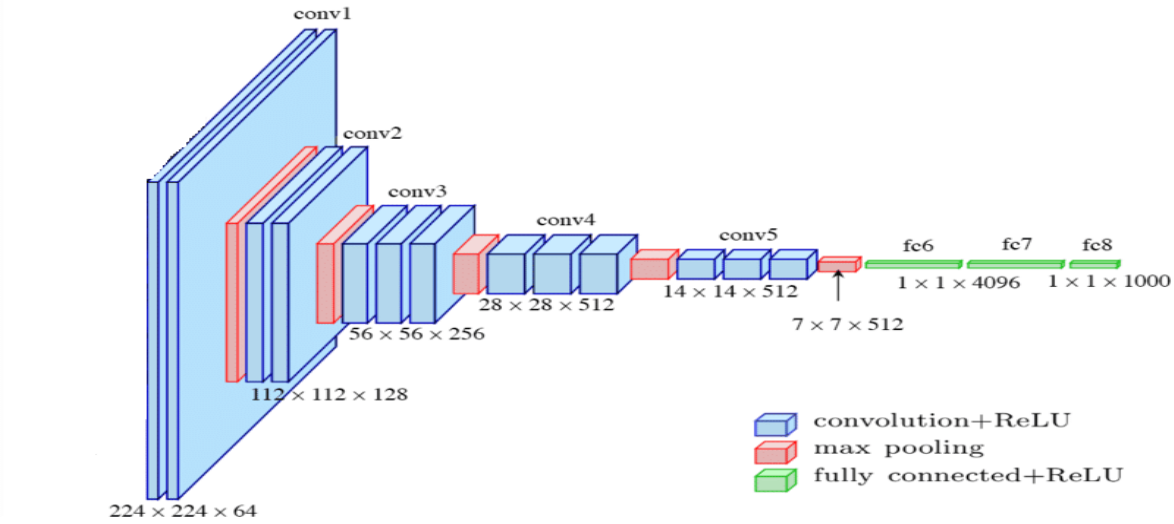

Figure 2.9: Standard VGG16 network architecture

VGG16 has 16 feature learning layers, 13 of which are convolution layers with 3x3 filters, 5 pooling layers, and three fully-connected layers where the first two have 4096 neurons and the final with 1000 neurons to produce a distribution over 1000 class labels. Each learnt layer is subjected to ReLU non-linearity.

The outcome shown in this network revealed that using a 3x3 convolutional filter acquired a significant advantage over using a 7x7 convolutional filter. The VGG-19, on the other hand, has three extra-convolution layers and a 3x3 receptive field.

Inception:

An Inception Network is a deep neural network that consists of repeating blocks where the output of a block acts as an input to the next block. Each block is defined as an Inception It employs numerous techniques to increase speed and precision of performance.

The Inception network was an essential milestone in the development of CNN classifiers which discovered how to reduce the computational cost of deep networks for classification without affecting the generalization. Inception-V1 which is another name for GoogleNet,

GoogLeNet achieved the state of the art performance in the classification computer vision task of the ImageNet Large Scale Visual Recognition Challenge 2014(ILVRC14). It successfully classified 1.2 million images into 1000 different classes, achieving a top-5 test error rate of 6.7%. This outperformed the second-best entry, which achieved a rate of 7.3% [42]. The key feature of GoogLeNet is its efficient utilization of computing resources within the network. It utilizes an efficient deep CNN architecture called Inception, which approximates and covers the optimal local sparse structure of a convolutional vision network using readily available dense components.

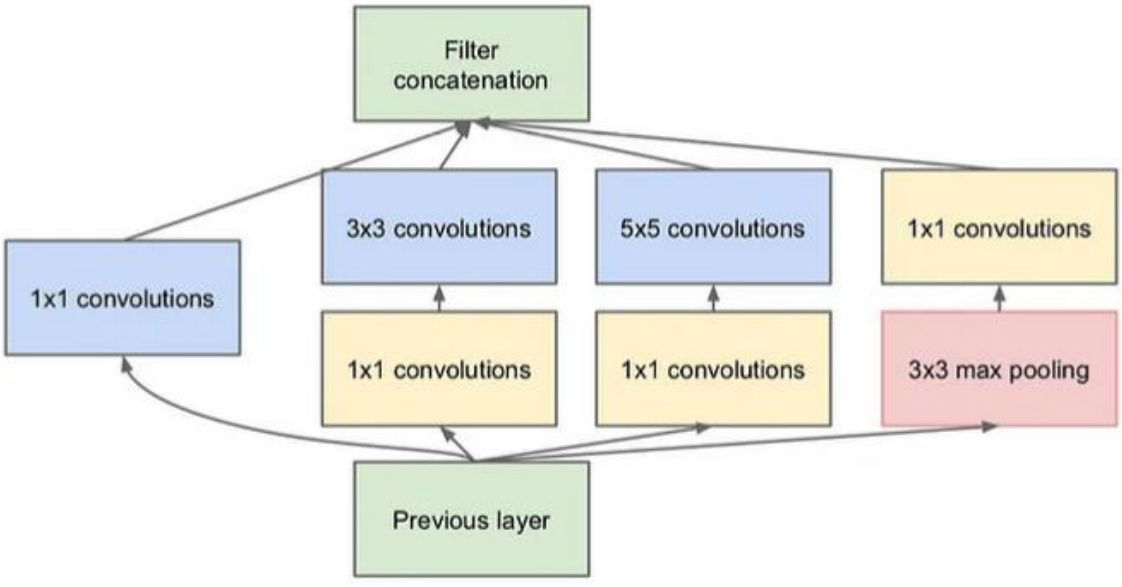

Figure 2.10: Inception model

The Inception model as shown in (figure 2.10), employed 1 x 1 convolutions to perform reductions prior to the computationally expensive 3 x 3 and 5 x 5 convolutions. These 1 x 1 convolutions served a dual purpose, functioning both as reductions and incorporating the use of ReLU activation. GoogLeNet consisted of 22 learned layers and 5 pooling layers. Instead of fully connected layers, it utilized average pooling, resulting in an improvement of approximately 0.6% in top-1 accuracy. All convolutions, including those within the Inception module, utilized ReLU non-linear activation. Inception-V2 is characterized by its notable uses of batch normalization, Inception-V3, V4, and InceptionResNet are other generations of the Inception network considered as improved versions of Inception-V1 and V2. Each version represents an incremental improvement over the one before it. Knowing the upgrades will enable us to create unique classifiers that are adjusted for both speed and precision.

ResNet:

ResNet (Residual networks) is the deepest network with 152 layers ever used on ImageNet, but it has less complexity than VGG. ResNet uses deep residual networks (ResNets) that speed up the training process and improve the accuracy compared to similar neural networks by adding a simple skip connection parallel to the convolutional layers. ResNets have shortcut connections that skip one or more layers and perform an identity mapping without adding extra parameters or complexity, and their outputs are added to the outputs of the stacked layers.

ResNet achieved the best performance in the ILSVRC-2015 classification competition, where it classified 1.2 million images into 1000 different classes with a top-5 test error rate of 3.57% [43]. It also won first places in ImageNet detection, ImageNet localization, COCO detection, and COCO segmentation tasks.

2.7. Related Work

In the literature review section, we attempted to explore and explain various research works conducted on identification or classification mechanisms in different domains. However, in our search, we also reviewed some research studies related to fish species identification, but to the best knowledge of the researcher, no research has been found for automatic fish species identification in Ethiopia. The following presents a selection of previous literature conducted by different researchers in this area.

2.7.1. Fish species identification

Several research works have been conducted on fish species identification using different machine learning approaches, image processing, and deep learning techniques.

The research work [44], developed a technique based on Convolutional Neural Networks, Deep Learning, and Image Processing for fish species classification. They used 21 species from internet images with a total image of 27,142. They got an accuracy of 96.29%. The method couldn't achieve 100% accuracy as some images couldn't be classified accurately due to the effect of background noise and other water bodies as the authors claimed.

The study [45], was on individual identification of fish without Obvious Patterns on the Body (Scale Pattern). It has used two fish species: the European seabass Dicentrarchus labrax and the common carp Cyprinus carpio. Four experimental data sets for each fish species were used to test the identification method, including two short-term (pattern variability test) and two long-term (pattern stability test) sets for European seabass (300 individual fish) and common carp (32 individual fish), used nearest-neighbor classifier. The classification accuracy was 100% for both fish species in both the short-term and long-term experiments. They tested only a maximum of 300 fish individuals. This study conducted a traditional way of image classification algorithm and used a very limited dataset.

The research [46] was on classification of fish species with augmented data using VGG16 deep convolutional neural network. The study used a total of 530 images for training and testing the model. This study used images of commonly found fish species in the Verde island passage. The model used 455 images for training and 75 images for testing purposes. A total fish image separated into three different fish species with a different number of training and testing images is applied. The model reached an overall accuracy of 99 %. The study used augmentation techniques but the data set is very small.

Study [47] developed a deep learning (DL)-based methodology for the recognition of different species of fish. Based on the YOLOv7 algorithm, a fish detection network (FD_Net) is proposed for this study. The dataset was split into three sections, which were designated as training, validation, and testing respectively. The performance of the FD_Net was examined and compared with YOLOv3, YOLOv3-TL & YOLOv3-BL, YOLOv4, YOLOv5, Faster-RCNN and the most recent YOLOv7. The total number of images used is 9000, where 6300 of it is for training, 1800 images for validation, and 900 images for testing. The proposed FD_Net model achieved the highest accuracy of 95.29%, YOLOv3 accuracy of 87.68%, YOLOv3-TL accuracy of 88.50%, YOLOv3-BL accuracy of 87.90%, YOLOv4 accuracy of 88.86%, YOLOv5 accuracy of 87.90%, Faster-RCNN and YOLOv7 accuracy of 85.76%, 90.42% respectively.

The study [48] used transfer learning to identify fish species based on the CNN architecture of VGG16 model. A total number of cropped images of 750, from 50 species each covered 15 images, 10 images for training purposes and 5 images for testing. The model trained on four different types of dataset: RGB color space image, canny filter image, blending image, and blending image mixed with RGB image. The results showed that the RGB color space image trained model had a genuine acceptance rate (GAR) value of 92.4%, while the blended image combined with RGB image trained model had the best GAR value of 96.4%, the canny filter image trained model with a genuine acceptance rate value of 80.4%, and the blending image trained model showed the least genuine acceptance rate value of 75.6%.

Study [49], proposed an enhanced version of the VGG-16 model for the automated classification of the underwater fish species and also proposed a technique based on the monitoring system of Fish Cam(video-based monitoring system). The data used in this study arrives from over three years of monitoring fish migration with the FishCam system across a variety of rivers in Austria. The approach is tested on a subset of 10 fish species (8099 individuals) occurring in the Austrian river. On an independent test set, the presented approach achieves a classification accuracy of 93%.

The research [50], was on the development of a prototype system named FishDeTec to detect the freshwater fish species found in Malaysia through the image processing approach. In this study, FishDeTec is developed using the VGG16 (deep Convolutional Neural Network (CNN) model) for large-scale image classification processing. The total number of images used in this study is 200 for 8 types of fish species. The model achieved 87% accuracy on the

validation dataset. The experimental result obtained in this study yielded a moderate performance as the accuracy is just between 60 to 80 percent throughout the 15 iterations. This may happen due to some factors such as the small size of the data set and the lack of image variation in every training sample.

The research [51], proposed an automatic fish species classification model based on deep convolutional neural networks. It uses a reduced version of the AlexNet model consisting of four convolutional layers and two fully connected layers. The QUT fish dataset was used as the source of the data for this study. QUT fish dataset contains 3960 images captured in different environments. In this research study, six fish species were selected and taken in different conditions. Only a total of 1344 images Used. The LifeClef2015 Fish dataset is used for testing purposes. The 20,000 images in this dataset were split up into 15 separate fish species classes. They have selected the same 6 fish species for training. The testing dataset was divided into testing and validation images. The testing images contain 20% of the images from the total number of images in each class. For validation, they randomly selected the images and took about 15% of the images from the total number of images in each class. Their proposed and modified AlexNet model with fewer layers has achieved a testing accuracy of 90.48% while the original AlexNet model achieved 86.65%.

The research [52], was on accurately identifying coral reef fishes in underwater images using CNN and comparing it to humans. First tested the performance of a CNN trained with different photographic databases while accounting for different post-processing decision rules to identify 20 fish species. Finally, the performance of species identification using CNN model compared with that of humans on a test database of 1197 fish images representing 9 species. With 900,000 images, CNN was trained. The rate of correct identification was 94.9%, and by humans (89.3%).

The study [53], proposed a hybrid Convolutional Neural Network (CNN) framework that uses CNN for feature extraction and Support Vector Machine (SVM) and K-Nearest Neighbour (k-NN) for classification. Both the proposed frameworks are tested on the Fish4Knowledge dataset. This dataset is acquired from the underwater observatories at Lake of Taiwan. The dataset contains a total image of 27,370. These fishes are divided into 23 species. The proposed framework Deep CNN-KNN gives an accuracy of 98.79%, DeepCNN accuracy of 98.65%, and DeepCNN-SVM accuracy of 98.32%.

The research [54], using YOLO deep learning. The dataset used in this research uses 24 species of reef fish. The test was carried out on 100 pictures containing 24 species to be identified. They got the overall accuracy of the model of 82.82%, and used a very small data set for a number of classes. In this study the model can identify fish but the results of identification do not match the actual fish species and the model cannot identify the fish in the picture.

The study [55], proposed using synthetic data to identify fish species in the absence of training data. Images were taken and acoustic data was gathered using acoustic-trawl surveys. They used a deep learning method (CNN) with training to simulate images realistically. The results showed a range of accuracy with a maximum accuracy of around 94 % for this approach.

The research [56], classified fish species using the K-Nearest Neighbor algorithm, on 160 datasets and 4 different fish species. Fish species are categorized using the K-NN approach using attributes such as color, texture, and shape. Using K = 7 and an accurate percentage of 77.50%, the accuracy value of the truth is calculated based on the test results. K = 10 has the second-highest accuracy value, which is 76.88%. According to the study's findings, the K-Nearest Neighbor approach can categorize fish reasonably well, but it can be improved by including additional factors or data and utilizing different kinds of fish.

Table 2.2: Summary of related works

Research title	**Authors**	**Algorithm's**	**Limitations**
Underwater Fish Species classification using Convolutional Neural Networks and deep learning	D.Rathi, S.Jain, & Dr. S. Indu, 2017	CNN	➢ There is background noise and other water bodies effects on the images, Enhancement techniques are not used well ➢ Dataset is not partitioned for Training, validating, and testing.
Classification of Fish Species with Augmented Data using Deep Convolutional Neural Network	Montalbo & Hernandez, 2019	VGG16	➢ Used traditional transformation techniques, Approved augmentation method like Keras is not used ➢ small sized dataset used ➢ Dataset size per class is not balanced

Image-Based automatic individual identification of Fish without obvious Patterns on the Body (Scale Pattern)	D.Bekkozhaye va, & P.Cisar, 2022	KNN HOG	➢ Used traditional machine learning algorithms ➢ Tested only maxi- mum of 300 fish individuals. ➢ very limited dataset ➢ Dataset is not partitioned for Training, validating, and testing
Fish Species Recognition Using VGG16 Deep Convolutional Neural Network	Hridayami, putra, & Wibawa, 2019	VGG16	➢ Dataset is not partitioned for training, validating, and testing ➢ Very limited dataset used; only 750 images for 50 species
Underwater Fish Species Recognition Using Deep Learning Techniques	B.V.Deep & R.Dash, 2019	CNN, DeepCNN-SVM, DeepCNN-KNN	➢ Used traditional machine learning algorithms for classification ➢ Dataset size per class is not balanced ➢ The model performance is not compared with state-of-the-art pre-trained models
Classification of Fish Species with Image Data Using K-Nearest Neighbor	K.Kaharuddin & E.W.Sholeha, 2021	KNN	➢ Very small dataset used; only 160 images ➢ Used traditional machine learning algorithms

2.7.2. Research Gap

Generally, we have reviewed research conducted on fish species identification using different approaches. While these studies have contributed valuable insights into the field, several research gaps can be identified that need to be addressed.

One prominent research gap is the limited dataset size used in many studies. The availability of a small number of fish species images for training and testing can impact the performance and generalization ability of the classification model. Therefore, addressing this research gap requires prioritizing the collection of larger and more diverse datasets to improve the accuracy and robustness of fish species identification models.

Additionally, the issue of imbalanced class distributions within the datasets has been observed. Some fish species are represented by a significantly larger number of samples compared to others, leading to biased models. To address this research gap, techniques for collecting and balancing datasets should be explored to ensure fair representation of different fish species. By tackling this issue, the overall accuracy of the fish species classification model can be enhanced.

Another research gap pertains to the impact of environmental factors on fish species identification. Several studies have acknowledged the presence of background noise and other water bodies as challenges that can hinder accurate classification. To overcome this limitation, robust techniques or preprocessing methods should be developed to mitigate the effects of environmental factors, thereby improving the accuracy of fish species identification in real-world aquatic environments.

Furthermore, most studies have primarily utilized traditional machine learning algorithms or relatively simple deep learning architectures for fish species identification. However, the potential of advanced deep learning techniques remains largely unexplored in this context. Therefore, using deep learning models and exploring the performance of state-of-the-art pre-trained models, such as ResNet, GoogLeNet, or Inception architectures, can help address this research gap and improve the accuracy and reliability of fish species identification.

In summary, the research gaps identified in the reviewed studies on fish species identification include the need for larger and more diverse datasets, addressing imbalanced class distributions, developing techniques to handle environmental factors, and exploring advanced deep learning models. By addressing these gaps, researchers can contribute to the advancement of accurate and robust fish species identification system.

CHAPTER 3: MATERIALS AND METHODS

3.1. Introduction

It takes a lot of research to identify an object and place it in the proper class. Each new object must be placed in one of the predefined classes in accordance with the observed qualities or features after passing through a sequence of stages or procedures that are done to differentiated items. The proposed model for automatic identification of fish species is explained in depth in this chapter. The sections of this chapter go into greater detail on the system model's constituent parts. Data acquisition, Pre-processing, Data augmentation, training and identification are the components. The architecture of the proposed model is shown in Figure 3.1 below.

3.2. Proposed System Architecture

The overall system model for identifying fish species consists of four components namely: acquisition, preprocessing, augmentation, training and identification.

The first component of the proposed model is acquisition. Fish species images were captured using smartphone camera. The images can be of different sizes, qualities, and angles.

In the second step, preprocessing, the images are normalized and enhanced to make them suitable for the next steps. The normalization involves resizing the images to a standard size and removing any noise or blur.

Deep learning is known for its huge demand for training data. In the third step, the data augmentation technique has been used to increase the diversity of training data. The images are transformed and modified to increase the diversity and size of the training data. The transformation involves applying random operations such as flipping, rotating, and zooming of the images.

In the fourth step, training and identification, the images are fed into a convolutional neural network (CNN) that learns to extract features and classify them into different fish species. The CNN model is trained on a large number of augmented images until it reaches a high accuracy. The model can then identify new images by using the learned features and labels.

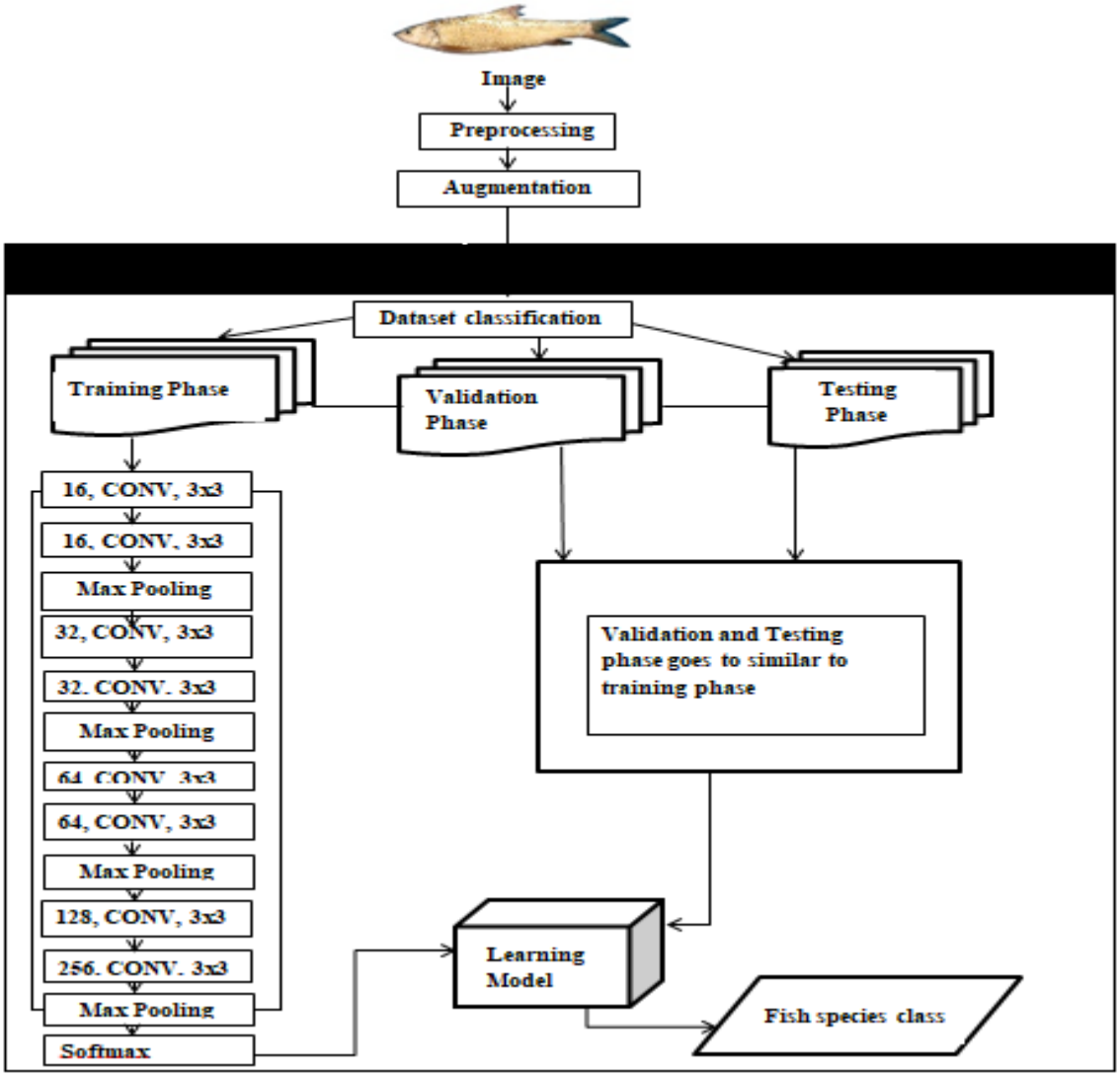

Figure 3.1: The architecture for automatic fish species identification

3.3. Dataset Preparation

Fish species samples were gathered from Lake Hawassa in Ethiopia. We created our own dataset to train on and assess performance because there is no ready-made dataset for this kind of research. The Center for Aquaculture Research and Education (CARE)'s domain experts, who are currently employed there, certified the labels of each species of fish. This study took into account a total of 6 fish species: *Oreochromis niloticus, Clarias garipienus, LabeoBarbus intermedius, Barbus paludinosis, Garra quadrimaculata* and *Aplocheilichthys*. Image acquisition is done using techno camon 18 smartphone camera having resolution of 48 Megapixels and it is loaded to the computer. The background color of the imaging system is nearly opposite to the fish color. For this study we collected a total of 3000 real images with 500 per class. While capturing each sample image we adjust the environment to gain uniform

lighting and we use a background object. Below Figure 3.2 are the sample fish species images of the six classes of the dataset.

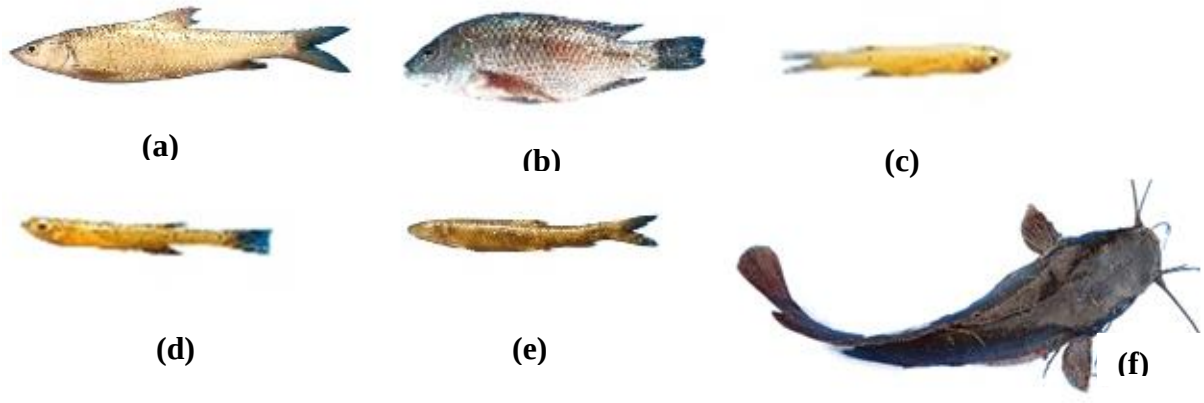

Figure 3.2: Sample of the fish species images collected.

(a) *LabeoBarbus intermedius* **(b)** *Oreochromis niloticus* **(c)** *Barbus paludinosis*
(d) *Aplocheilichthys* **(e)** *Garra quadrimaculata* **(f)** *Clarias garipienus*

3.4. Preprocessing

Image pre-processing is the common name for an operation with images at the lowest abstract level. The input and output are the intensity of the image. Most image processing techniques involve noise reduction, contrast enhancement, image resizing, color correction, segmentation, feature extraction. The input image is pre-processed by the pre-processing component.

The second component of the proposed model is the preprocessing module. The preprocessing module is responsible for making the images ready for the overall image analysis process. The preprocessing module for this study mainly contains three main components such as image resizing, pixel value normalization and In order to remove noise from the image we have used median filtering technique.

3.4.1. Image Resizing

Image resizing is a preprocessing technique that makes a number of images to similar size. Images gathered from various sources may vary in size. Therefore, it should be resized and scaled to fit an equivalent sized image. As working with larger images size is computationally very expensive so it is very important. Before data can be put into the network, it must be sized and formatted appropriately. We have chosen 224x224 image sizes that best reflect the image content with less processing time and also it aligns with the

commonly used input size in state-of-the-art models [40] [42]. By utilizing a similar image size, we can readily evaluate our network by comparing its performance against these well-established models. This allows for a fair assessment and enables us to determine the effectiveness of our approach in comparison to recognized benchmarks. The 224x224 size captures fine-grained features and larger patterns, essential for accurate classification or object detection. The 224x224 image size is highly preferred due to its widespread adoption, compatibility, efficiency, and ability to capture important image characteristics. All images in our dataset are resized to 224 x 224 pixels according to Algorithm 3.1.

Algorithm 3.1: Image resizing

Input : Image
Output : Resized Image
Begin : For image in images path Resized Image=resize (image, (224,224)) Return Resized image
End

3.4.2. Pixel Value Normalization

Another preprocessing method we utilize before additional processing is normalization. The integer values for image pixels fall between 0 and 255. Although the model can be given these pixel values directly, doing so may cause overflow and delayed training. When numbers become too large and the computer cannot do the computation correctly, overflow occurs. Therefore, by dividing the pixel values by 255, we apply Algorithm 3.2 to normalize our data values down to a decimal between 0 and 1.

Algorithm 3.2: Normalization

Input : Resized Image pixel value(pixel value)
Output : Normalized Image pixel value between (0,1)
value= 255 Begin : For image in dataset Normalized_Image = pixel value /value Return Normalized_Image
End

3.5. Data Augmentation

In order to expose the model to a greater variety of training examples, image augmentation involves applying changes to images to produce different versions of the same content. For the greatest results, the CNN architecture must be trained on a large number of images for each class. Applying augmentation primarily serves to increase the dataset's volume by slightly distorting the images. A CNN model's capacity to learn more precise features is increased by having a large dataset. This will reduce overfitting and improve training performance. It also enables the model to generalize effectively to testing samples that have not been seen.

Deep learning neural network library Keras is used to augment data for transformation, random rotation, horizontal and vertical flip through ImageDataGenerator class. This layer allows applying data enhancement and artificially creating different variations of the same image. We apply Algorithm 3.3 to augment our data to artificially generate different variations of the same image.

Algorithm 3.3: Augmentation

Input : preprocessed image
Output : augmented image
Begin : For image in dataset datagen = ImageDataGenerator(rotation_range=15, width_shift_range=0.2, heignt_shift_range=0.2, zoom_range=0.1, shear_range=0.2, horizontal_flip=True) Return datagen
End

We applied random transformation (such as rotation, shearing, flipping and resizing) to our dataset to balance the insufficiency of our data. It may reduce training accuracy since the input images are changed slightly as compared to training without data augmentation. In addition, it enables the model to generalize better to new input samples.

Figure 3.3: The original input image (right) Effect of data augmentation on the original image (left)

3.6. Fish species Identification

The process of classification involves the automatic learning of distinguishing characteristics during the training phase. In feature learning, various layers are built on top of one another according to the defined architectural layout. The Fish species classifier architecture building involves the training, validation, and testing phases. In the following section, a description of the proposed architecture and operations for each phase that is used for learning features and classifying the species images into predefined 6 classes is presented.

Training Model

Deep learning techniques like CNN's overcome the difficulty of manual feature extraction in traditional machine learning algorithms by learning them automatically while training the network with sample images. In our study we used 70/30 and 80/20 dataset ratios. 70% of our dataset is used to train the model, the remaining 30% of the datasets are used for testing the performance of the trained model and 80% of our dataset is used to train the model, the remaining 20% of the datasets are used for testing respectively. We selected the 80/20 and 70/30 partition ratios because the 80/20 ratio provides a larger portion for training the model, allowing it to learn from a substantial amount of data. With our relatively small dataset, allocating 80% for training enables the model to capture underlying patterns effectively and potentially improve its performance. Additionally, we considered the 70/30 ratio to strike a balance between training and testing data. By assigning 70% for training, we ensured the model had access to a significant amount of data while reserving 30% for evaluating its performance on unseen samples. This ratio provides a substantial testing set while still allowing enough training data to capture complex patterns. It is important to note that our

choice of partition ratios was based on general considerations and assumptions about the dataset and model complexity. We also took into account previous experiences and empirical evidence that indicated the effectiveness of allocating 2/3rd of the dataset for training [57].

Convolutional neural network (CNN) architectures based on two methods are designed to classify different fish species, namely: Training using CNN architectures and Training using pre-trained models. This section describes the CNN architectures which are designed for this study using these two methods.

3.6.1. Training using convolutional neural networks (CNN)

The convolutional neural network is trained to classify each species, and the identification is done after learning the distinguishing features. Feature learning consists of different layers stacked on top of each other. A fish species dataset was created to identify different fish species. A convolutional neural network contains convolutional layers, pooling layers, fully connected layers, and others. Each layer description and proposed architecture of the CNN classifier model is described in this section.

In our model there is convolutional neural network that has 224 x 224 RGB fish species image dimension and channel, respectively and training parameters such as number of filters, filter size, stride and padding. We have used five numbers of filters with 3 x 3 filter size, 1x1 strides size and padding the same. We have used different activation functions namely Rectified Linear Unit (ReLU) for feature extraction and Softmax for classification. After the convolution layer, we applied a pooling operation to reduce the number of parameters and computational load in our model by reducing the input image. It reduces each feature map by downsampling to control overfitting and preserve the most important information. Max pooling is an operation to calculate the maximum values and average pooling is a pooling operation to calculate the average values in each part of the feature map. In our model we have used a Max pooling layer. The stride and pool size are two parameters in pooling activity. At the end of convolution/pooling (previous layers) operation the last output of pooling operation is used as the input of a fully connected layer. Fully connected layer performs identification task based on the feature extracted in convolution pooling operation. To identify each different fish species we have used the Softmax classifier.

As shown in Figure 3.4, the first and the second convolution layer has 16 kernels with a size of 3 x 3 filters and a max pooling layer with a 2 x 2 window. In the third and the fourth convolution layer, we have used 32 kernels with 3 x 3 size of filter and a max pooling layer with a 2 x 2 window size.

In the fifth and the sixth convolution layer, 64 kernels with 3 x 3 filter size, a max pooling layer with a 2 x 2 window size and 0.25 dropout layer. In the seventh convolution layer, we have used 128 kernels with 3 x 3 filter size. In the last convolution layer, we have used 256 kernels with 3 x 3 size of filter, a max pooling layer with a 2 x 2 window size and we add dropout layer 0.25. Each convolution layer is followed by nonlinear activation function (ReLU). It is the common activation function which is faster than sigmoid function; due to this it reduce the training time accordingly.

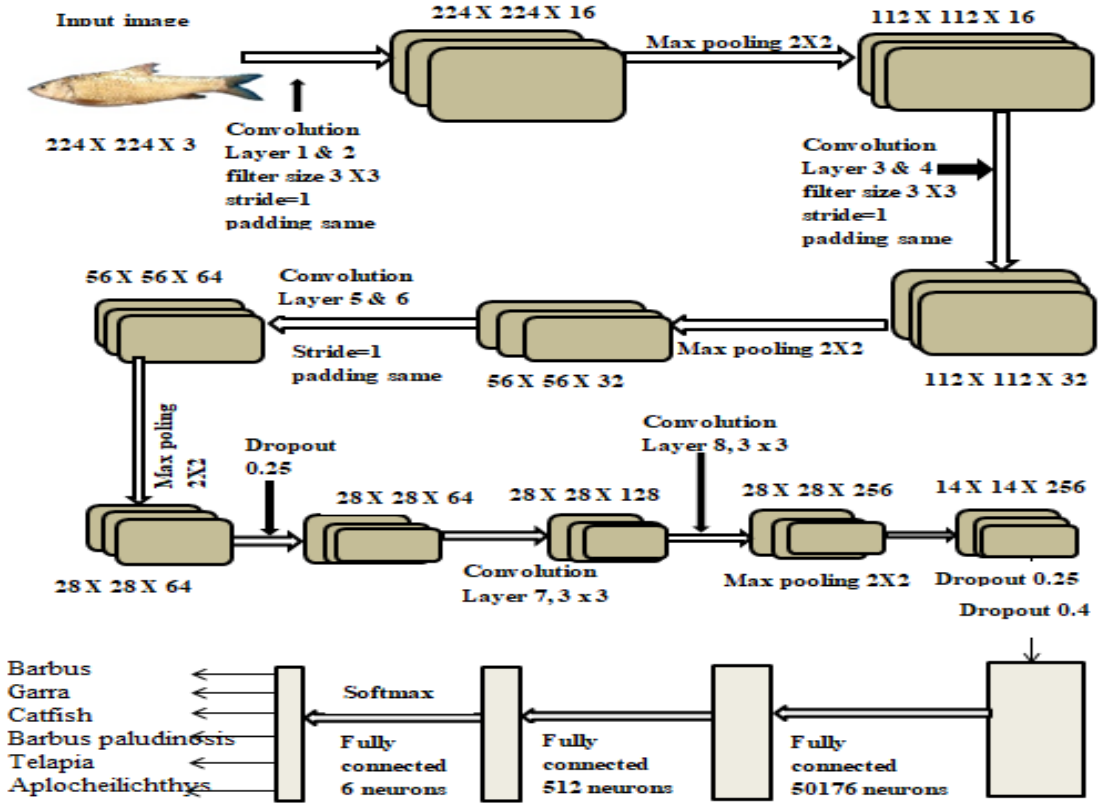

Figure 3.4: Classification model architecture of the proposed model

3.6.2. Training using pre-trained models

We used three state-of-the-art pre-trained models that were trained on a different and much larger dataset. Such as, Inceptionv3 network, VGG16, and Residual Network (ResNet50). These models are discussed in chapter 2 section 2.6 and are general models for computer vision tasks that were trained on ImageNet dataset. We did not need to train their network from scratch, as we could use their learned feature maps from Keras as a starting point. We modified the final classification layer of the base convolutional network to match our number of classes and fine-tuned the higher-level features of the base model to make them more relevant for our specific task. We also adjusted some parameters such as batch size, learning rate, and epochs for the training.

CHAPTER 4: RESULTS AND DISCUSSION

4.1. Introduction

This chapter discusses experimental evaluation of the proposed model for fish species identification. Experimental evaluation is made to evaluate the effectiveness of our proposed architecture or how the model performs well. The dataset used and the implementation results obtained from different evaluation techniques are described. Comparison results are presented in training using Convolutional neural network (CNN) architectures with two dataset ratios and using pretrained models for fish species identification.

4.2. Data Collection and Preparation

The first activity of a deep learning based classification system is data collection and Preparation. In this experiment, there is no ready-made dataset, so we have collected and prepared our own data set from scratch. The sample data of fish species collected from Hawassa Lake.

The collected sample data of fish species are certified by domain experts who are working in the CARE (Centre for Aquaculture Research and Education) in Hawassa University. The data set was captured using techno camon 18 mobile phone with different dimensions. For this study we used a total of 6000 images dataset with 1000 per class.

To evaluate the performance of the model, we have used Evaluation metrics for two different ratios of dataset namely 70/30 and 80/20 and we used three pre trained models (VGG16, Inceptionv3 and ResNet50). Table 4.1 shows the detailed description of the dataset that we have used in this experiment.

Table 4.1: Dataset Description

No	Species Name	Image Format	Data source	No of Collected Data	After Augmentation
1	*LabeoBarbus intermedius*	JPG	Lake Hawassa	500	1000
2	*Clarias garipienus*	JPG	Lake Hawassa	500	1000
3	*Oreochromis niloticus*	JPG	Lake Hawassa	500	1000
4	*Garra quadrimaculata*	JPG	Lake Hawassa	500	1000
5	*Barbus paludinosis*	JPG	Lake Hawassa	500	1000
6	*Aplocheilichthys*	JPG	Lake Hawassa	500	1000

4.3. Development environment

Experiments are done based on Keras with a TensorFlow backend on Intel(R) Core(TM) i5-7200U 2.50GHz 2.70GHz CPU, 8 GB of RAM with a 64-bit Microsoft Windows 10 operating system for preprocessing and using Google Colab which allows us to have free access to NVIDIA Tesla T4 GPUs and 13 GB RAM for 12 hours per session.

In this study, we have used supporting tools such as Keras, Tensorflow. Tensorflow is a powerful library that makes deep learning faster. Keras is user friendly and provides certain python libraries and other python libraries that enable developers to use optimized algorithms and implements popular machine learning techniques in classification.

The module 'tf.keras.preprocessing.image.ImageDataGenerator' in tensorflow is used to aid the augmentation, providing operations that are typically used in the generation of image data for deep learning problems. Some of the important python libraries used in this study are NumPy, Pandas and Matplotlib.

4.4. Implementation

The proposed model is trained for 50 epochs, a batch size of 64, and a starting or initial learning rate of 0.0001 (1e-4). The data is partitioned into two ratios 70/30, 80/20 for training and testing dataset. 70 percent of the data is assigned for training the model and 30 percent of the data is allocated for testing, 80 percent of the data is assigned for training the model and 20 percent of the data is allocated for testing respectively.

4.5. Performance evaluation and Testing

In this study we have followed different scenarios to evaluate the performance of the fish species identification proposed model. To measure the performance of our model we have used accuracy, recall, precision and f1-score. These metrics are dependent on the Confusion Matrix which contains important information about the model. It is an n x n matrix that contains the number of occurrences of actual value and predicted value.

Table 4.2: Confusion matrix table

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

True Positives (TP): These are the correctly predicted positive as positive which means that the value of the actual class is positive and the value of the predicted class is also positive.

True Negatives (TN): These are correctly predicted as negative values which mean that the value of the actual class is negative and the value of the predicted class negative.

False Positives (FP): This occurs when the classifier predicted a positive outcome while the actual class was negative.

False Negatives (FN): The classifier predicts a negative class when the real class is positive.

Accuracy: Overall performance of the model.

$$\text{Accuracy} = (TP + TN) / (TP + T\,N + FP + FN)$$

Recall: Coverage of actual positive samples.

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

Precision: How the positive description is accurate.

$$\text{Precision} = TP / (TP+FP)$$

F1-Score: is the weighted average between precision and recall.

$$\text{F1-score} = 2 * (\text{precision} * \text{recall}) / (\text{precision} + \text{recall})$$

4.5.1. Hyperparameter setting of fish species identification model using CNN

In our experiment, 6 Fish species images of 224x224 size with the associated class label were fed into the proposed CNN architecture, this architecture has 8 convolutional layers with 5 filters (16, 32, 64, 128 and 256), 4 Max-pooling layers, 2 dropout layers that ignore 25% and 40% of the output layer, and one fully-connected layers with 512 hidden neurons and the final classification layer six classes.

We have used 50 training epochs to train our model. Since the batch size is 64, the network parameter updates every time with a block of 64 images as shown in Table 4.3 below. It also describes the detailed hyperparameter setting of CNN model to identify 224x224 image sizes of six different fish species.

The performance of CNN model depends on the image size that we have used. If the image size and the dataset size become large it takes more training time. The architecture of the model has 8 convolutional layers followed by a max pooling layer.

The Rectified Linear Unit (ReLU) activation function is used after every convolution operation and at the end the extracted feature vector enters the fully connected layer to compute the final output of the identification.

Since we have six classes, the final output layer contains six nodes. In this output layer we have used the Softmax activation function as a classifier.

The detailed hyperparameter configuration of CNN model that we have used in our model is illustrated below Table 4.3. We have used a 70/30 ratio which means 70% of the data is for training and 30% of the data is allocated for testing.

Table 4.3: Hyperparameter setting used in CNN model training

Input: Image Size (224 × 224)				
Conv1: 3X3	Activation : ReLU	stride:1	padding: same	Feature maps: 16
Conv2: 3X3	Activation : ReLU	stride:1	padding: same	Feature maps: 16
Conv3: 3X3	Activation : ReLU	stride:1	padding: same	Feature maps: 32
Conv4: 3X3	Activation : ReLU	stride:1	padding: same	Feature maps: 32
Conv5: 3X3	Activation : ReLU	stride:1	padding: same	Feature maps: 64
Conv6: 3X3	Activation : ReLU	stride:1	padding: same	Feature maps: 64
Conv7: 3X3	Activation : ReLU	stride:1	padding: same	Feature maps: 128
Conv8: 3X3	Activation : ReLU	stride:1	padding: same	Feature maps: 256
Dropout: 0.25, Dropout: 0.4 Batch size: 64 Hidden units: 512 Classifier: Softmax Number of class: 6 Optimizer: Adam(Learning rate: 0.0001) Loss: categorical_crossentropy Epochs: 50				

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 224, 224, 16)	448
conv2d_1 (Conv2D)	(None, 224, 224, 16)	2320
max_pooling2d (MaxPooling2D)	(None, 112, 112, 16)	0
conv2d_2 (Conv2D)	(None, 112, 112, 32)	4640
conv2d_3 (Conv2D)	(None, 112, 112, 32)	9248
max_pooling2d_1 (MaxPooling2D)	(None, 56, 56, 32)	0
conv2d_4 (Conv2D)	(None, 56, 56, 64)	18496
conv2d_5 (Conv2D)	(None, 56, 56, 64)	36928
max_pooling2d_2 (MaxPooling2D)	(None, 28, 28, 64)	0
dropout (Dropout)	(None, 28, 28, 64)	0
conv2d_6 (Conv2D)	(None, 28, 28, 128)	73856
conv2d_7 (Conv2D)	(None, 28, 28, 256)	295168
max_pooling2d_3 (MaxPooling2D)	(None, 14, 14, 256)	0
dropout_1 (Dropout)	(None, 14, 14, 256)	0
flatten (Flatten)	(None, 50176)	0
dense (Dense)	(None, 512)	25690624
dropout_2 (Dropout)	(None, 512)	0
dense_1 (Dense)	(None, 6)	3078

Figure 4.1: Summary of the proposed CNN model

In CNN model training, we have done two experimentations by varying the amount of dataset used in model but the same hyper-parameter setting (as shown in Table 4.3). In the first experimentation, we have used a total of 3000 dataset and 70/30 ratio for training and testing. We have used 1200 fish species images for training and 900 for validation (as shown Figure 4.2) in the training and validation Accuracy/loss curve.

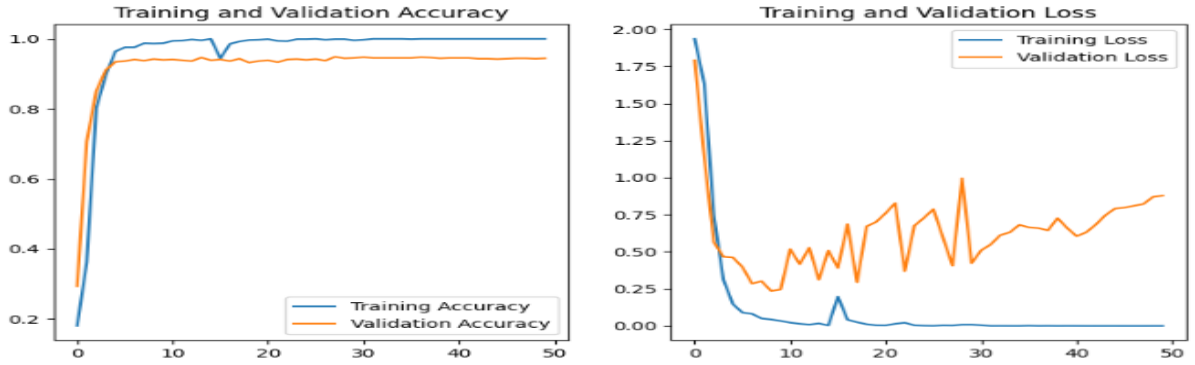

Figure 4.2: Training and validation Accuracy/loss curve with small dataset

```
29/29 [==============================] - 4s 135ms/step - loss: 1.5269 - accuracy: 0.9400
```

Figure 4.3: Test result of CNN Model with small dataset

To evaluate this CNN model we have used confusion matrix, precision, recall, and F1-score performance measurement technique. The following diagram below shows the confusion matrix table to understand how well our model performs prediction. In this experiment a confusion matrix with a 6x6 matrix is used to evaluate the performance of our model. This matrix compares the actual value with predicted value. It is obtained from the CNN model with six classes of fish species. As shown in Figure 4.4, the predicted labels are listed along the x-axis and the true label also listed along the y-axis. The correct identifications are shown along in the first diagonal. But all other entries show misidentification.

Figure 4.4: Confusion matrix of the proposed CNN model with small dataset

In Figure 4.5 below false positive rate has been shown by precision, false negative has shown by recall, f1-score shows the average or the harmonic mean of precision and recall.

	precision	recall	f1-score	support
0	0.88	0.95	0.91	150
1	0.87	0.95	0.90	150
2	0.94	0.94	0.94	150
3	0.99	0.93	0.96	150
4	0.99	0.94	0.97	150
5	1.00	0.93	0.97	150
accuracy			0.94	900
macro avg	0.94	0.94	0.94	900
weighted avg	0.94	0.94	0.94	900

Figure 4.5: Precision, recall, and F1-score of CNN model

In the second experimentation we have used a total 6000 fish species image dataset. Since the dataset size is larger than in the first experimentation, the gap between the validation and training accuracy is narrow as discussed in below. This indicates that the good performance of this CNN model.

4.6. Performance Evaluation of the Proposed Model with 70/30 Ratio

The proposed model was trained using a 70/30 ratio, with 70% of the dataset allocated for training and 30% for testing. The model utilized CNN architecture and was trained with specific parameters, including 50 epochs, a batch size of 64, a dropout rate of 0.25 and 0.4 and Adam optimizer with a learning rate of 0.0001. During the training process, the model achieved a training accuracy of 100%, indicating a strong fit to the training data. The validation accuracy reached 99.7%, demonstrating good generalization to unseen data. When evaluated on the testing set, the model achieved an accuracy of approximately 99.5%, indicating its high performance in predicting fish species.

To assess the model's performance, various performance measurement techniques were employed. The confusion matrix, represented in Figure 4.9, compared the actual and predicted labels for the six classes of fish species. The diagonal of the matrix represented correct predictions, while off-diagonal entries indicated incorrect predictions. It was observed that the model correctly predicted the majority of the fish species classes.

Additionally, Figure 4.10 displayed the metrics of precision, recall, and F1-score. Precision quantified the false positive rate, measuring the ratio of correctly predicted positive instances to the total predicted positive instances. Recall quantified the false negative rate, measuring the ratio of correctly predicted positive instances to the total actual positive instances. The F1-score, the harmonic mean of precision and recall, provided an overall measure of the model's classification performance.

In summary, the model trained with a 70/30 ratio demonstrated high accuracy on the test set and performed well for most fish species classes. However, only three classes had a smaller misclassification rate, with small percent values of instances being incorrectly predicted as another class.

```
Epoch 1/50
38/38 [==============================] - 583s 15s/step - loss: 1.7903 - accuracy: 0.1988 - val_loss: 1.7581 - val_accuracy: 0.5189
Epoch 2/50
38/38 [==============================] - 14s 356ms/step - loss: 1.2920 - accuracy: 0.5696 - val_loss: 0.8202 - val_accuracy: 0.7072
Epoch 3/50
38/38 [==============================] - 13s 353ms/step - loss: 0.6722 - accuracy: 0.7383 - val_loss: 0.3905 - val_accuracy: 0.8683
Epoch 4/50
38/38 [==============================] - 13s 350ms/step - loss: 0.3727 - accuracy: 0.8637 - val_loss: 0.2254 - val_accuracy: 0.9261
Epoch 5/50
38/38 [==============================] - 15s 393ms/step - loss: 0.2504 - accuracy: 0.9146 - val_loss: 0.1832 - val_accuracy: 0.9389
                                        .                    .              .
                                        .                    .              .
                                        .                    .              .
Epoch 45/50
38/38 [==============================] - 13s 350ms/step - loss: 0.0034 - accuracy: 0.9992 - val_loss: 0.0222 - val_accuracy: 0.9961
Epoch 46/50
38/38 [==============================] - 13s 339ms/step - loss: 0.0051 - accuracy: 0.9992 - val_loss: 0.0186 - val_accuracy: 0.9956
Epoch 47/50
38/38 [==============================] - 13s 333ms/step - loss: 0.0098 - accuracy: 0.9987 - val_loss: 0.0213 - val_accuracy: 0.9939
Epoch 48/50
38/38 [==============================] - 13s 345ms/step - loss: 0.0097 - accuracy: 0.9979 - val_loss: 0.0370 - val_accuracy: 0.9922
Epoch 49/50
38/38 [==============================] - 13s 348ms/step - loss: 0.0052 - accuracy: 0.9996 - val_loss: 0.0368 - val_accuracy: 0.9944
Epoch 50/50
38/38 [==============================] - 13s 352ms/step - loss: 0.0096 - accuracy: 0.9971 - val_loss: 0.0255 - val_accuracy: 0.9944
```

Figure 4.6: Training process of CNN model using 70/30 ratio dataset

Figure 4.7 represents the accuracy and loss graph for the 70/30 ratio. It shows the progress of the model's training over the course of the epochs. The x-axis represents the number of epochs, while the y-axis represents the training accuracy. From the figure, it can be observed that the training accuracy gradually increases as the number of epoch's increases. Eventually, it reaches around 100% accuracy, indicating that the model has learned the training data well.

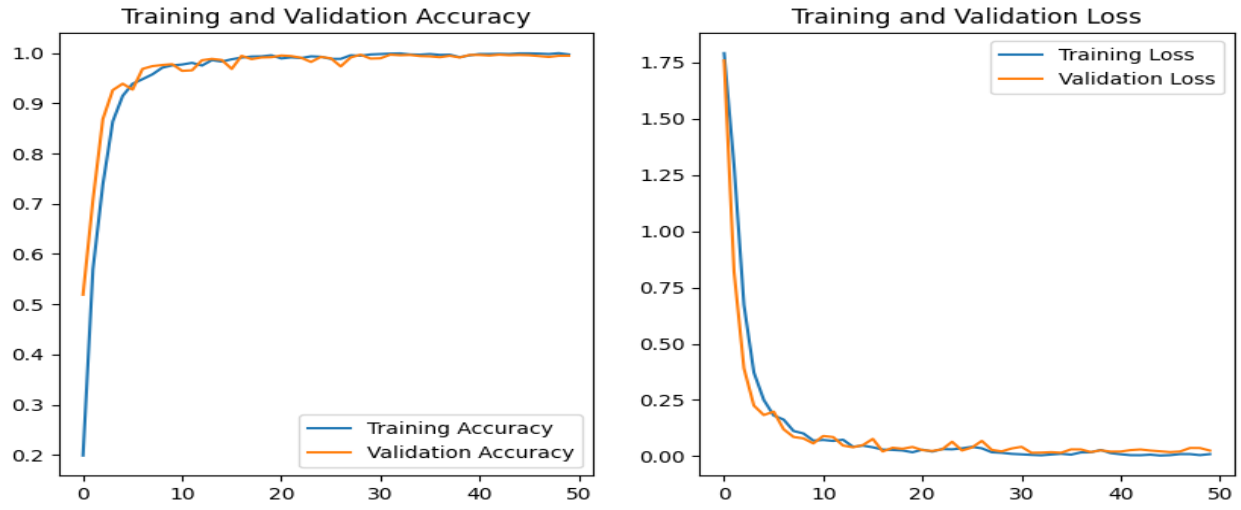

Figure 4.7: Accuracy and loss of Training and validation

```
57/57 [==============================] - 305s 5s/step - loss: 0.0225 - accuracy: 0.9956
Test Loss: 0.02252795547246933
Test Accuracy: 0.995555579662323
```

Figure 4.8: Testing result of CNN model

The normalized confusion matrix shown in Figure 4.9 below can be calculated by dividing each row element by the total of the entire row and this final normalized confusion matrix shows the percentage. In this experiment a confusion matrix with a 6x6 matrix is used to evaluate the performance of our model.

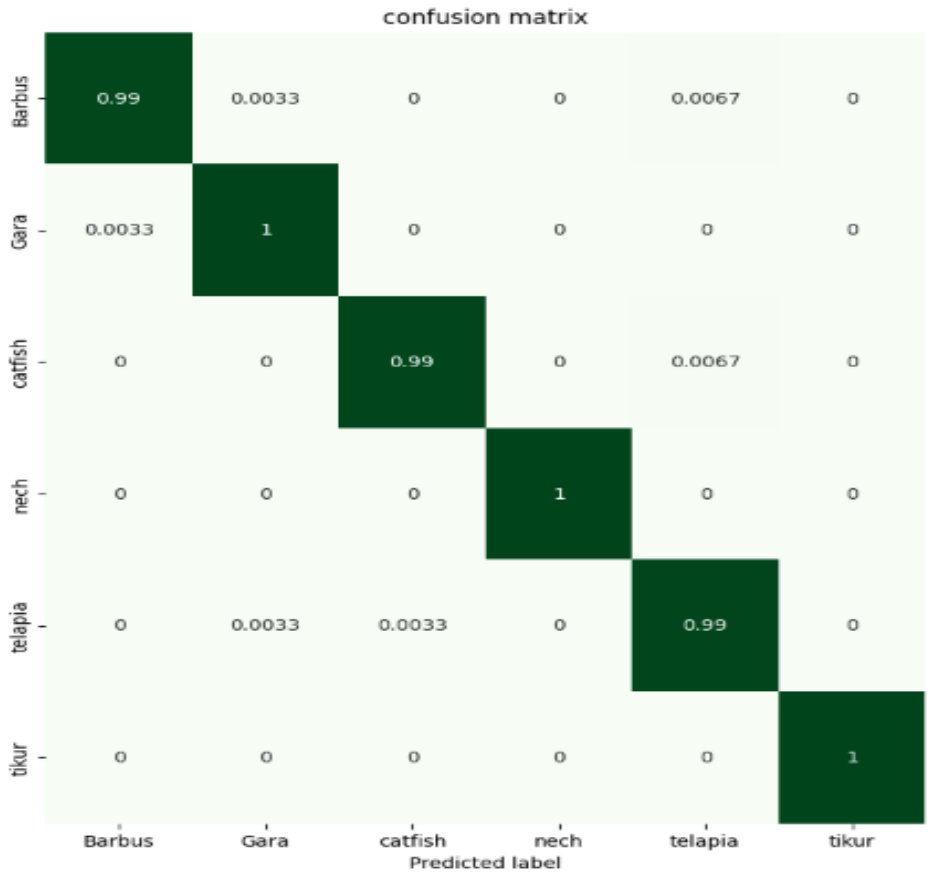

Figure 4.9: Confusion matrix of the proposed CNN model

This matrix compares the actual value with predicted value. It is obtained from the CNN model with six classes of fish species. As shown in the Figure, the predicted labels are listed along the x-axis and the Actual label also listed along the y-axis. The correct identifications are shown along in the diagonal. But all other entries show incorrect identifications.

As shown in the confusion matrix (Figure 4.9) above, Barbus fish species samples are 99% correctly predicted. Whereas, 0.33% of the sample is incorrectly predicted as Garra and 0.67% of the Barbus fish species sample is also incorrectly predicted as Telapia.

In the second row of the matrix, Garra fish species samples are 99.9% correctly predicted whereas, 0.33% of the sample is incorrectly predicted as Barbus.
In the third row of the matrix, Catfish fish species samples are 99% correctly predicted whereas 0.67% of the sample is incorrectly predicted as Telapia. In the fifth row of the matrix, Telapia fish species samples are 99% correctly predicted whereas, 0.33% of the sample is incorrectly predicted as Garra and 0.33% of the Telapia fish species sample is also incorrectly predicted as Catfish. As shown in the Figure confusion matrix it is easier to interpret the predicted value of each label.

Figure 4.10 below illustrates the evaluation metrics such as precision, recall, and F1-score. Precision represents the false positive rate, which measures the ratio of correctly predicted positive instances to the total predicted positive instances. Recall represents the false negative rate, which measures the ratio of correctly predicted positive instances to the total actual positive instances. The F1-score is the harmonic mean of precision and recall, providing a balanced measure of the model's performance. By analyzing these metrics, we can assess the model's ability to correctly classify instances and balance false positives and false negatives.

	precision	recall	f1-score	support
0	1.00	0.99	0.99	300
1	0.99	1.00	1.00	300
2	1.00	0.99	0.99	300
3	1.00	1.00	1.00	300
4	0.99	0.99	0.99	300
5	1.00	1.00	1.00	300
accuracy			1.00	1800
macro avg	1.00	1.00	1.00	1800
weighted avg	1.00	1.00	1.00	1800

Figure 4.10: Precision, recall, and F1-score of CNN model

4.7. Performance Evaluation of the Proposed Model with 80/20 Ratio

In the second experiment, a performance evaluation was conducted on the proposed model with an 80/20 ratio. The dataset was divided into 80% for training and 20% for testing. The model utilized CNN architecture with specific parameters, including 50 epochs, a batch size of 64, Adam optimizer with a learning rate of 0.0001, and a dropout rate of 0.25 and 0.4. During the training process, the model achieved an impressive training accuracy of 99.8%, indicating a strong fit to the training data. The validation accuracy was equally impressive

reaching 99.9%, demonstrating the model's ability to generalize well to unseen data. When evaluated on the testing set, the model achieved an accuracy of approximately 98.3%, indicating its high performance in predicting fish species.

```
Epoch 1/50
57/57 [==============================] - 20s 290ms/step - loss: 1.7014 - accuracy: 0.2769 - val_loss: 1.0541 - val_accuracy: 0.6192
Epoch 2/50
57/57 [==============================] - 18s 311ms/step - loss: 0.7044 - accuracy: 0.7247 - val_loss: 0.4436 - val_accuracy: 0.8458
Epoch 3/50
57/57 [==============================] - 16s 274ms/step - loss: 0.3625 - accuracy: 0.8683 - val_loss: 0.2136 - val_accuracy: 0.9175
Epoch 4/50
57/57 [==============================] - 18s 309ms/step - loss: 0.2502 - accuracy: 0.9089 - val_loss: 0.1906 - val_accuracy: 0.9208
Epoch 5/50
57/57 [==============================] - 16s 277ms/step - loss: 0.1633 - accuracy: 0.9453 - val_loss: 0.1236 - val_accuracy: 0.9525
                                        .                    .                .
                                        .                    .                .
                                        .                    .                .
Epoch 45/50
57/57 [==============================] - 18s 307ms/step - loss: 0.0154 - accuracy: 0.9964 - val_loss: 0.0366 - val_accuracy: 0.9975
Epoch 46/50
57/57 [==============================] - 16s 271ms/step - loss: 0.0160 - accuracy: 0.9964 - val_loss: 0.0267 - val_accuracy: 0.9967
Epoch 47/50
57/57 [==============================] - 16s 280ms/step - loss: 0.0042 - accuracy: 0.9994 - val_loss: 0.0217 - val_accuracy: 0.9975
Epoch 48/50
57/57 [==============================] - 16s 274ms/step - loss: 0.0021 - accuracy: 0.9994 - val_loss: 0.0103 - val_accuracy: 0.9975
Epoch 49/50
57/57 [==============================] - 18s 311ms/step - loss: 0.0027 - accuracy: 0.9992 - val_loss: 0.0277 - val_accuracy: 0.9975
Epoch 50/50
57/57 [==============================] - 16s 277ms/step - loss: 0.0037 - accuracy: 0.9986 - val_loss: 0.0019 - val_accuracy: 0.9992
```

Figure 4.11: Training process of CNN model using 80/20 ratio dataset

Figure 4.12 represents the training process for the 80/20 ratio. It shows the progress of the model's training over the course of the epochs. The x-axis represents the number of epochs, while the y-axis represents the training accuracy. From the figure, it can be observed that the training accuracy gradually increases as the number of epoch's increases. Eventually, it reaches around 99.86% accuracy, indicating that the model has learned the training data well.

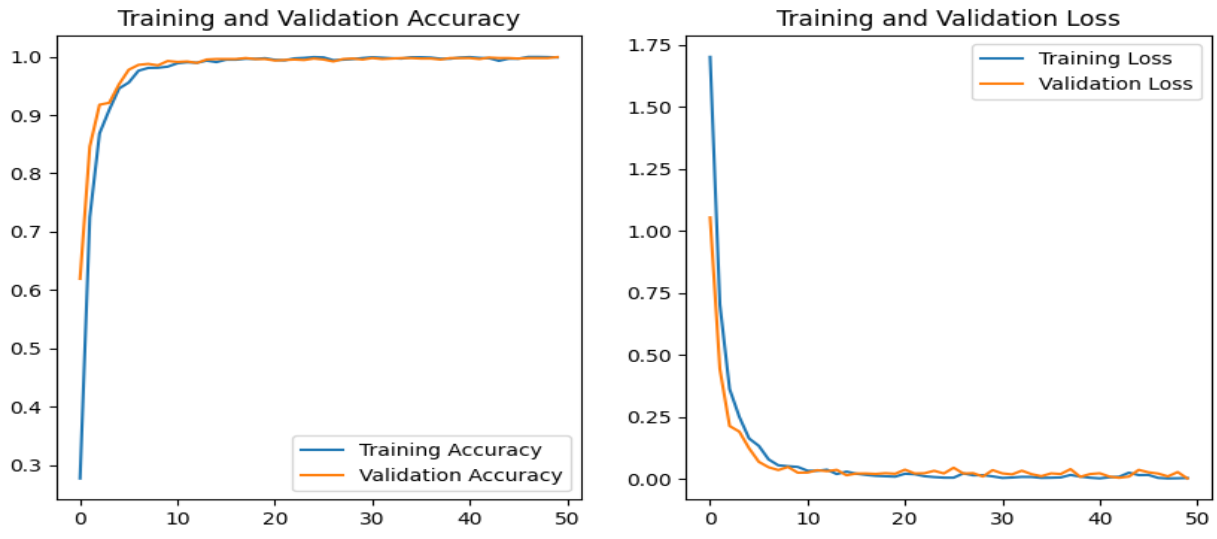

Figure 4.12: Accuracy and loss plot of 80/20 ratio

```
38/38 [==============================] - 50s 1s/step - loss: 1.3914 - accuracy: 0.9833
Test Loss: 1.391444444656372
Test Accuracy: 0.9833333492279053
```

Figure 4.13: Testing result of CNN model with 80/20 ratio

To evaluate the model's performance, various performance measurement techniques were employed. The confusion matrix, depicted in Figure 4.15, compared the actual and predicted labels for the six classes of fish species. The diagonal entries of the matrix represented correct predictions, while the off-diagonal entries indicated incorrect predictions. It was observed that, except for one class, the model correctly predicted the majority of the fish species classes. However, one specific class, known as Barbus, exhibited a slightly higher misclassification rate, with some percentage of instances being incorrectly predicted as another class. This finding suggests that further analysis and investigation are necessary to understand and address the challenges faced by the model in accurately predicting this particular class.

	precision	recall	f1-score	support
0	1.00	0.90	0.95	200
1	1.00	1.00	1.00	200
2	0.92	1.00	0.96	200
3	1.00	1.00	1.00	200
4	1.00	1.00	1.00	200
5	1.00	1.00	1.00	200
accuracy			0.98	1200
macro avg	0.98	0.98	0.98	1200
weighted avg	0.98	0.98	0.98	1200

Figure 4.14: Precision, recall, and F1-score of CNN model with 80/20

Furthermore, Figure 4.14 above provided additional insights into the model's performance. The metrics of precision, recall, and F1-score were displayed in the figure. Precision measured the false positive rate, which is the ratio of correctly predicted positive instances to the total predicted positive instances. These metrics allowed for a comprehensive assessment of the model's predictive capabilities and provided valuable information for evaluating its performance across different classes.

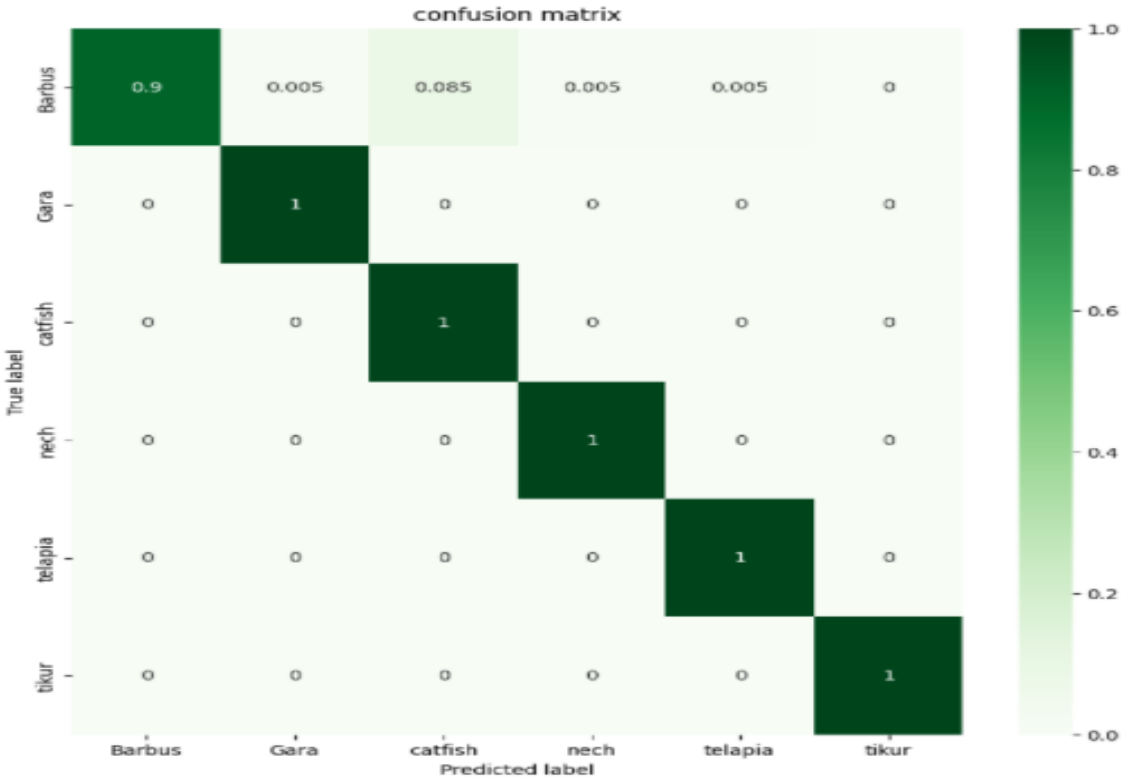

Figure 4.15: Confusion matrix of the model with 80/20

In conclusion, the model trained with an 80/20 ratio achieved high accuracy on the test set and demonstrated strong performance for most fish species classes. However, further analysis is recommended to address the challenges faced by the model in accurately predicting the Barbus class and to potentially improve its performance in future experiments.

4.8. Comparative Analysis of Model Performance with 80/20 and 70/30 Data Ratios

This summary presents a detailed comparative analysis of the performance of a proposed model trained with two different data ratios: 80/20 and 70/30. The objective is to assess how the allocation of training and testing data impacts the model's accuracy, training, testing, and F1-score.

The model trained with a 70/30 ratio (Experiment 4.6) demonstrated superior performance compared to the model trained with an 80/20 ratio (Experiment 4.7). Experiment 4.6 achieved remarkable accuracy metrics, with a training accuracy of 100%, a validation accuracy of 99.7%, and an accuracy of approximately 99.5% on the testing set. In contrast, Experiment 4.7 achieved slightly lower accuracy metrics, with a training accuracy of 99.8%, a validation accuracy of 99.9%, and an accuracy of approximately 98.3% on the testing set.

Both experiments utilized the confusion matrix to assess the model's predictions for different fish species classes.

In Experiment 4.6, the model correctly predicted the majority of the fish species classes. Similarly, Experiment 4.7 achieved accurate predictions for most fish species classes, with one class exhibiting a slightly higher misclassification rate. To provide a comprehensive evaluation, precision, recall, and F1-score metrics were employed. Although specific values were not mentioned, it can be inferred that Experiment 4.6 achieved better results in terms of these metrics compared to Experiment 4.7, aligning with the higher overall accuracy and lower misclassification rate observed in Experiment 4.6.

In conclusion, the comparative analysis of the proposed model's performance with 80/20 and 70/30 data ratios highlights the superiority of the model trained with a 70/30 ratio. Experiment 4.6 demonstrated better accuracy, training, testing, and F1-score compared to Experiment 4.7. However, both models faced challenges in accurately predicting a specific fish species class, indicating the need for further investigation and enhancement. These findings contribute valuable insights into choosing an optimal data ratio for achieving superior performance in similar classification tasks.

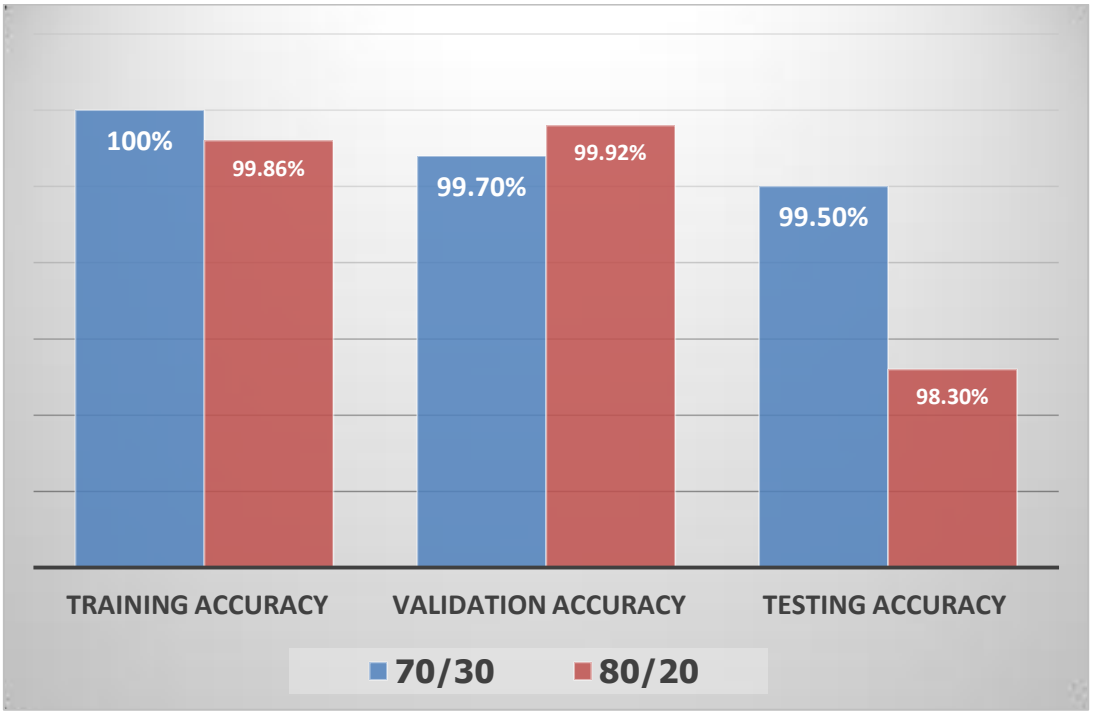

Figure 4.16: Comparison of Model Performance with 80/20 and 70/30 Data Ratios

4.9. Evaluation of the Proposed Model with Pretrained Models

To measure how well our model performs, we compare it with the best models that have won ILSVRC (ImageNet Large Scale Visual Recognition Challenges) in different years. These models are VGGNet, GoogLeNet, and ResNet, and they are explained in detail in the literature review section (chapter two).

We have trained and tested each model using our own dataset. We evaluate the proposed model with pretrained models using those parameters such as Loss of the model and Accuracy of the model. We trained those three pre-trained models on a dataset of 6000 images of six types of fish species: Barbus, Telapia, Catfish, Garra, nech, and tikur.

We used the proposed 70/30 dataset split ratio for training and testing purposes. We use categorical cross-entropy as the loss function, which measures the difference between the predicted probabilities and the true labels and Adam as the optimizer. We use a batch size of 64, which means that we feed 64 images at a time to the model and update the weights after each batch. We train those three pre-trained models for 50 epochs like we used for the proposed model, which means that we go through the entire training set 50 times.

4.9.1. Comparison with VGG16 Model

We use the VGG16 model, which is a pre-trained convolutional neural network (CNN) that we download from 'keras.applications.VGG16'. We add three more dense layers with 512, 256, and 128 neurons to the existing classification layers. We only train 1,214,086 parameters out of the total 23,016,870 parameters in the model. Then we apply a flattening operation and add three more dense layers with 512, 256, and 128 neurons each and 50 % Dropout layer. The last layer is a prediction layer that has six neurons, which is the number of classes in our dataset. We train the model and achieve 98.04% training accuracy and 95% validation accuracy.

Figure 4.17 below shows the training process and the performance (Training accuracy and Training loss value, Validation accuracy and Validation loss value) of the VGG16 model and clearly shows, VGG16 achieves 98.04% training and 95.60% testing accuracy on our data.

```
Epoch 1/50
38/38 [==============================] - 95s 1s/step - loss: 0.9771 - accuracy: 0.6779 - val_loss: 0.2024 - val_accuracy: 0.9383
Epoch 2/50
38/38 [==============================] - 41s 1s/step - loss: 0.2631 - accuracy: 0.9087 - val_loss: 0.1191 - val_accuracy: 0.9589
Epoch 3/50
38/38 [==============================] - 42s 1s/step - loss: 0.1526 - accuracy: 0.9496 - val_loss: 0.1249 - val_accuracy: 0.9606
Epoch 4/50
38/38 [==============================] - 41s 1s/step - loss: 0.1359 - accuracy: 0.9592 - val_loss: 0.1008 - val_accuracy: 0.9667
Epoch 5/50
38/38 [==============================] - 43s 1s/step - loss: 0.1060 - accuracy: 0.9646 - val_loss: 0.0733 - val_accuracy: 0.9800
                                        .                          .              .
                                        .                          .              .
                                        .                          .              .
Epoch 45/50
38/38 [==============================] - 42s 1s/step - loss: 0.0187 - accuracy: 0.9925 - val_loss: 0.0329 - val_accuracy: 0.9872
Epoch 46/50
38/38 [==============================] - 42s 1s/step - loss: 0.0234 - accuracy: 0.9933 - val_loss: 0.0015 - val_accuracy: 1.0000
Epoch 47/50
38/38 [==============================] - 40s 1s/step - loss: 0.0246 - accuracy: 0.9921 - val_loss: 0.0050 - val_accuracy: 0.9989
Epoch 48/50
38/38 [==============================] - 45s 1s/step - loss: 0.0258 - accuracy: 0.9925 - val_loss: 0.0293 - val_accuracy: 0.9883
Epoch 49/50
38/38 [==============================] - 42s 1s/step - loss: 0.0198 - accuracy: 0.9946 - val_loss: 0.0045 - val_accuracy: 0.9983
Epoch 50/50
38/38 [==============================] - 42s 1s/step - loss: 0.0222 - accuracy: 0.9912 - val_loss: 0.1170 - val_accuracy: 0.9572

57/57 [==============================] - 349s 6s/step - loss: 0.1195 - accuracy: 0.9560
Test Loss: 0.11950640380382538
Test Accuracy: 0.955988883972168
```

Figure 4.17: Training process of VGG16 Model

	precision	recall	f1-score
0	1.00	1.00	1.00
1	1.00	1.00	1.00
2	1.00	1.00	1.00
3	1.00	0.74	0.85
4	1.00	1.00	1.00
5	0.79	1.00	0.88
accuracy			0.96
macro avg	0.97	0.96	0.96
weighted avg	0.97	0.96	0.96

Figure 4.18: Classification report of VGG16 Model

As shown in the Figure 4.18 (Classification report of VGG16 Model) training with VGG16 results in the correct classification of 100% of Barbus, African Catfish and nech, Telapia and tikur fish species and 74 % of Garra fish species are correctly classified. In general, 95% testing accuracy was recorded in this training.

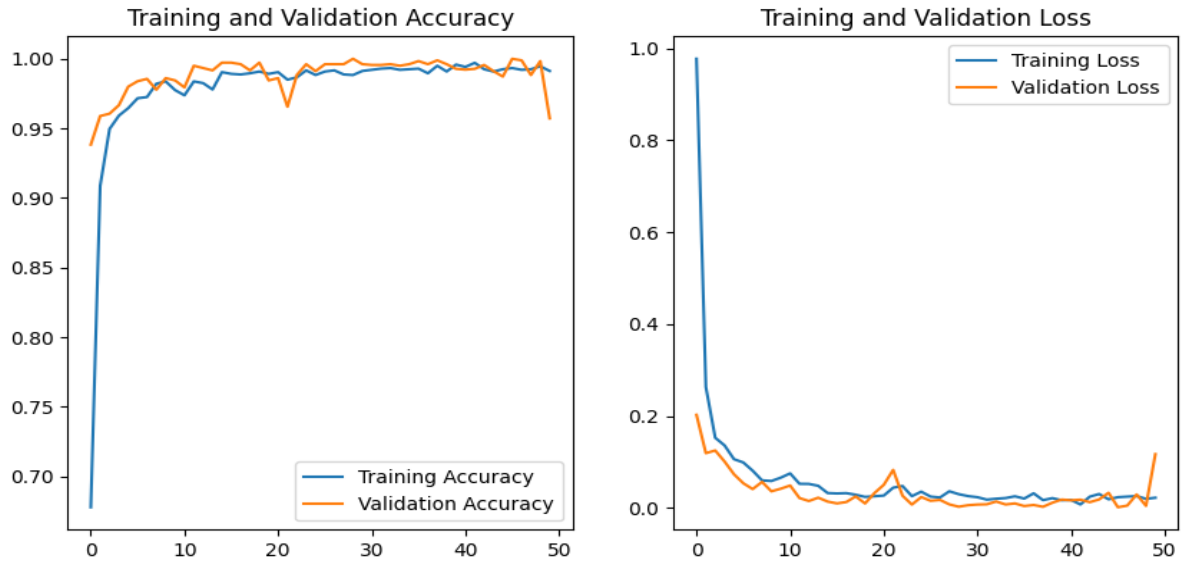

Figure 4.19: VGG16 Model accuracy and loss plot

4.9.2. Comparison with Inceptionv3 Model

On top of the pre-trained Inceptionv3 model downloaded from 'keras.applications. Inceptionv3'. We extend the classification layers by adding 3 dense layers 512, 256 and 128. Out of a total of 23,016,870 parameters, only 1,214,086 parameters were trained.

Then after adding a flattening operation followed by 3 dense layers having 512,256 and 128 neurons and the final prediction layer, which has the length of our classes that is 6, we trained the network and obtained 99 % training accuracy and 99 % validation accuracy.

The performance (Training accuracy and Training loss value, Validation accuracy and Validation loss value) of Inceptionv3 model is shown in figure 4.20 below. Inceptionv3 obtains 99 % training and 98.72% testing accuracy on our data.

```
Epoch 1/50
38/38 [==============================] - 1081s 28s/step - loss: 0.6726 - accuracy: 0.7708 - val_loss: 0.1523 - val_accuracy: 0.9678
Epoch 2/50
38/38 [==============================] - 46s 1s/step - loss: 0.1680 - accuracy: 0.9550 - val_loss: 0.1475 - val_accuracy: 0.9467
Epoch 3/50
38/38 [==============================] - 41s 1s/step - loss: 0.1249 - accuracy: 0.9571 - val_loss: 0.2342 - val_accuracy: 0.8972
Epoch 4/50
38/38 [==============================] - 40s 1s/step - loss: 0.0977 - accuracy: 0.9688 - val_loss: 0.1399 - val_accuracy: 0.9428
Epoch 5/50
38/38 [==============================] - 41s 1s/step - loss: 0.0911 - accuracy: 0.9704 - val_loss: 0.0800 - val_accuracy: 0.9694
                                        .                          .              .
                                        .                          .
Epoch 45/50
38/38 [==============================] - 40s 1s/step - loss: 0.0124 - accuracy: 0.9954 - val_loss: 0.0238 - val_accuracy: 0.9917
Epoch 46/50
38/38 [==============================] - 41s 1s/step - loss: 0.0086 - accuracy: 0.9967 - val_loss: 0.0184 - val_accuracy: 0.9883
Epoch 47/50
38/38 [==============================] - 41s 1s/step - loss: 0.0171 - accuracy: 0.9954 - val_loss: 0.0197 - val_accuracy: 0.9894
Epoch 48/50
38/38 [==============================] - 40s 1s/step - loss: 0.0238 - accuracy: 0.9929 - val_loss: 0.0668 - val_accuracy: 0.9700
Epoch 49/50
38/38 [==============================] - 45s 1s/step - loss: 0.0079 - accuracy: 0.9983 - val_loss: 0.0583 - val_accuracy: 0.9756
Epoch 50/50
38/38 [==============================] - 40s 1s/step - loss: 0.0143 - accuracy: 0.9946 - val_loss: 0.0074 - val_accuracy: 0.9983

57/57 [==============================] - 477s 9s/step - loss: 0.1645 - accuracy: 0.9872
Test Loss: 0.1645093560218811
Test Accuracy: 0.9872221946716309
```

Figure 4.20: Training process of Inceptionv3 Model.

	precision	recall	f1-score	support
0	1.00	1.00	1.00	300
1	0.94	1.00	0.97	300
2	1.00	1.00	1.00	300
3	1.00	0.97	0.98	300
4	1.00	0.93	0.97	300
5	0.97	1.00	0.98	300
accuracy			0.98	1800
macro avg	0.98	0.98	0.98	1800
weighted avg	0.98	0.98	0.98	1800

Figure 4.21: Classification report of Inceptionv3 Model

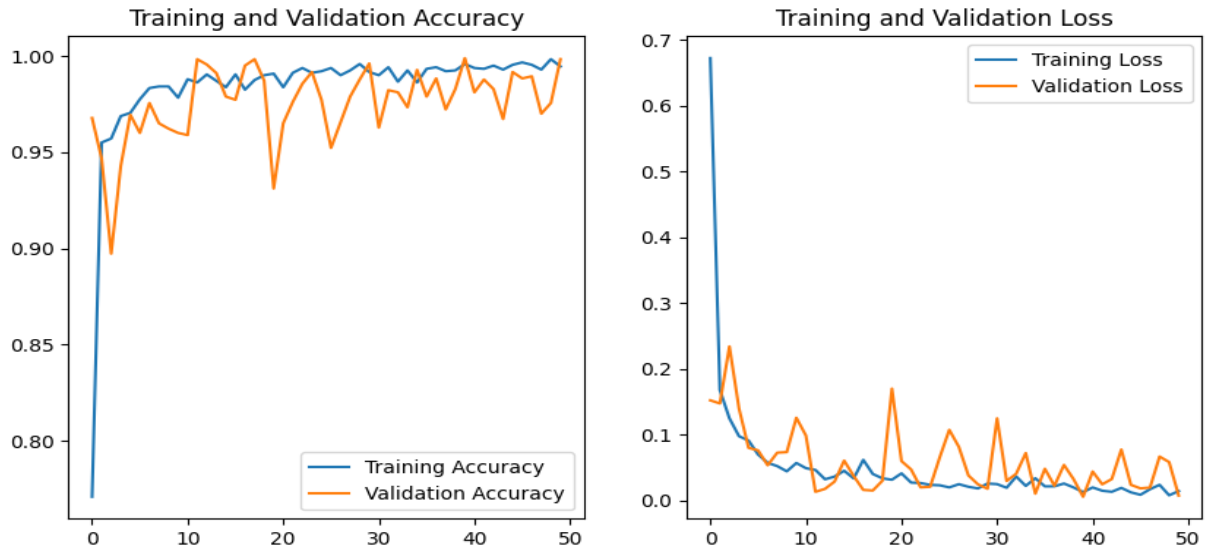

Figure 4.22: Inceptionv3 Model accuracy and loss plot

As shown in the Figure 4.22 (Classification report of Inceptionv3 Model) training with Inceptionv3 results in the correct classification of 100% of Barbus, Africa Catfish, tikur fish species, 97% nech fish species and 93 % of Telapia fish species are correctly classified. In general, 98% testing accuracy was recorded in this training.

4.9.3. Comparison with ResNet50 Model

The performance (Training accuracy and Training loss value, Validation accuracy and Validation loss value) of the ResNet50 model is shown in figure 4.18 below. ResNet50 obtains 98.75% training and 98.33% testing accuracy on our data. It is lower (about 2% down) than our model, which obtains 100 % training and 99% testing accuracy.

On top of the pre-trained ResNet50 model downloaded from 'keras.applications.ResNet50'. We extend the classification layers by adding 1 dense layer. Out of a total of 24,639,878 parameters, only 1,050,166 parameters were trained.

Then after adding a flattening operation followed by 1 dense layer having 512 neurons and the final prediction layer, which has the length of our classes that is 6, we trained the network and obtained 98.75% training accuracy and 97.44% validation accuracy.

```
Epoch 1/50
38/38 [==============================] - 3493s 93s/step - loss: 1.4826 - accuracy: 0.4512 - val_loss: 1.1148 - val_accuracy: 0.6189
Epoch 2/50
38/38 [==============================] - 14s 376ms/step - loss: 0.9828 - accuracy: 0.7138 - val_loss: 0.8589 - val_accuracy: 0.8294
Epoch 3/50
38/38 [==============================] - 19s 496ms/step - loss: 0.8026 - accuracy: 0.7954 - val_loss: 0.7485 - val_accuracy: 0.7156
Epoch 4/50
38/38 [==============================] - 14s 372ms/step - loss: 0.6970 - accuracy: 0.8117 - val_loss: 0.6661 - val_accuracy: 0.8433
Epoch 5/50
38/38 [==============================] - 14s 373ms/step - loss: 0.6044 - accuracy: 0.8512 - val_loss: 0.5636 - val_accuracy: 0.8622
                                        .                       .             .
                                        .                       .             .
                                        .                       .             .
Epoch 45/50
38/38 [==============================] - 14s 367ms/step - loss: 0.0898 - accuracy: 0.9837 - val_loss: 0.1113 - val_accuracy: 0.9750
Epoch 46/50
38/38 [==============================] - 14s 365ms/step - loss: 0.0857 - accuracy: 0.9858 - val_loss: 0.1171 - val_accuracy: 0.9689
Epoch 47/50
38/38 [==============================] - 14s 365ms/step - loss: 0.0847 - accuracy: 0.9862 - val_loss: 0.0978 - val_accuracy: 0.9761
Epoch 48/50
38/38 [==============================] - 14s 366ms/step - loss: 0.0848 - accuracy: 0.9854 - val_loss: 0.1030 - val_accuracy: 0.9744
Epoch 49/50
38/38 [==============================] - 14s 370ms/step - loss: 0.0852 - accuracy: 0.9846 - val_loss: 0.1217 - val_accuracy: 0.9728
Epoch 50/50
38/38 [==============================] - 14s 366ms/step - loss: 0.0811 - accuracy: 0.9846 - val_loss: 0.0972 - val_accuracy: 0.9744

57/57 [==============================] - 1380s 25s/step - loss: 0.0811 - accuracy: 0.9833
Test Loss: 0.08106754720211029
Test Accuracy: 0.9833333492279053
```

Figure 4.23: Training process of ResNet50 Model

	precision	recall	f1-score	support
0	1.00	1.00	1.00	300
1	0.99	0.92	0.95	300
2	1.00	1.00	1.00	300
3	0.99	1.00	0.99	300
4	1.00	0.99	0.99	300
5	0.93	0.99	0.96	300
accuracy			0.98	1800
macro avg	0.98	0.98	0.98	1800
weighted avg	0.98	0.98	0.98	1800

Figure 4.24: Classification report of ResNet50 Model

Figure 4.24 shows the training with ResNet50 results in the correct classification of 100% of Barbus, African Catfish and nech fish species and 92 % of Garra fish species, 99% of Telapia and tikur are correctly classified. In general, 98% testing accuracy was recorded in this training.

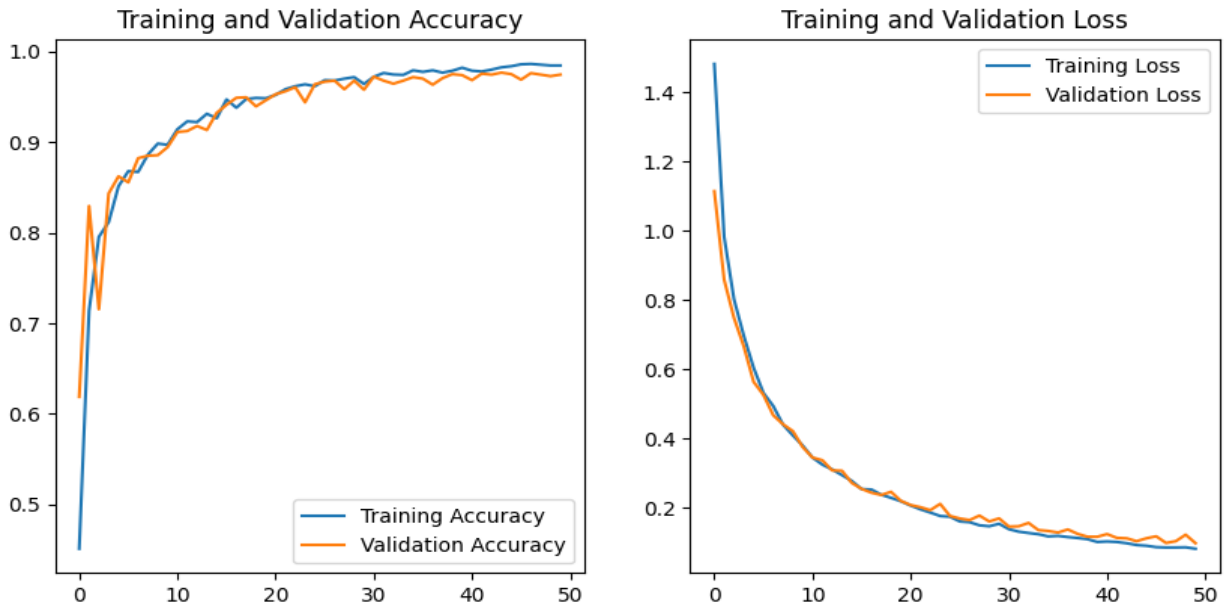

Figure 4.25: ResNet50 Model accuracy and loss plot

4.9.4. Summary of Comparison

Our proposed model based on the CNN architecture outperformed the VGG16, GoogLeNetv3 (Inceptionv3), and ResNet50 models in terms of training, validation, and testing accuracy in the task of fish species classification.

Compared to VGG16, our model achieved a higher testing accuracy of 99% compared to VGG16's 95.60%. It also achieved higher accuracy rates for individual fish species, including Barbus, Africa Catfish, nech, Telapia, tikur, and Garra.

In the comparison with Inceptionv3, our model demonstrated exceptional performance with a testing accuracy of 99.5% compared to Inceptionv3's 98%. The proposed model had high accuracy in both training and testing, indicating its effectiveness in accurately classifying fish species.

Furthermore, our model outperformed the ResNet50 model, achieving a higher testing accuracy of 99% compared to ResNet50's 98.33%. This suggests that our proposed model is better suited for accurately classifying the fish species in the dataset.

Overall, our proposed model based on the CNN architecture showed superior performance and higher accuracy in fish species identification compared to the VGG16, Inceptionv3, and ResNet50 models.

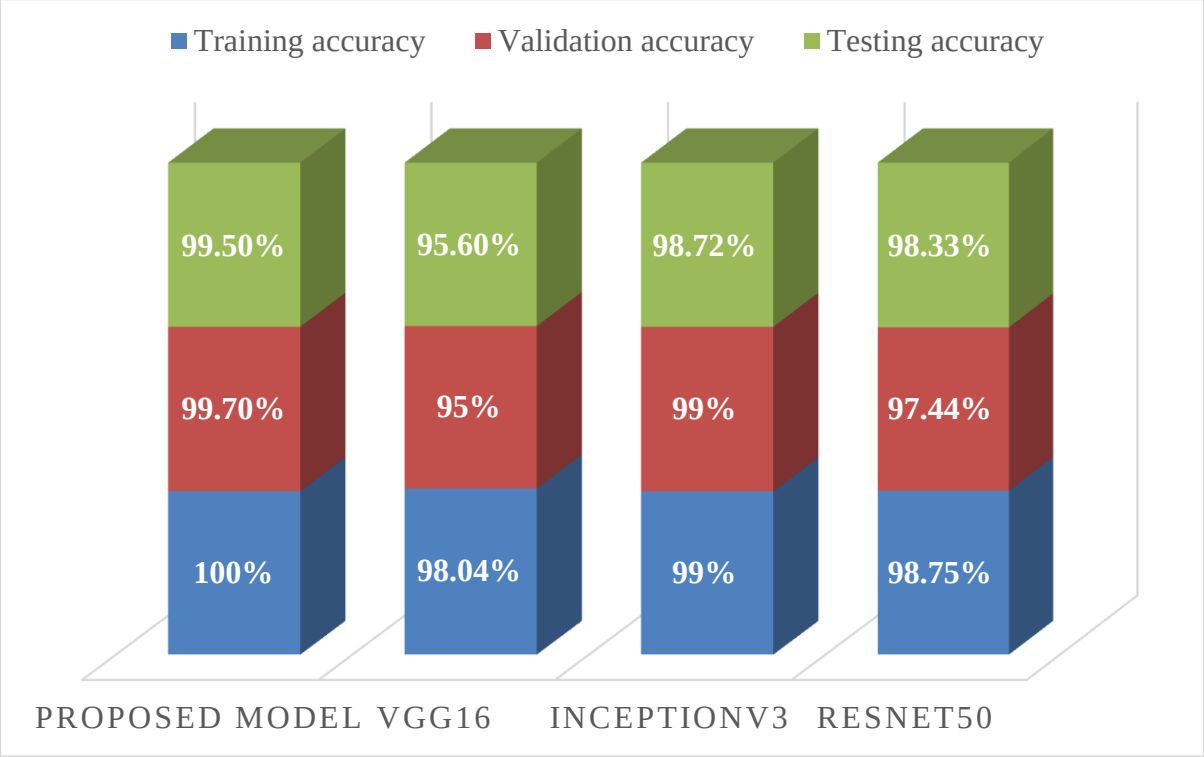

Figure 4.26: Comparative Performance Analysis of Pre-trained Models and Proposed Model for Fish Species Identification

4.9.5. Comparison of the Proposed Model with Related Works

In this section, we compared the performance of our proposed model with some of the related works that attempted to classify different fish species, as discussed in chapter 2, section 2.7. We selected the related works that used traditional machine learning and deep learning techniques for fish species classification. We describe the comparison of our proposed model with those works is tabulated in the following Table 4.4, based on the classifier approach and the accuracy achieved.

Table 4.4: Comparison of the proposed model with related works

No.	Authors	Classifier	Accuracy
1	D.Rathi, S.Jain, & Dr. S. Indu [44]	CNN	96.29%
2	B.V.Deep & R.Dash [53]	SVM	98.32%
		KNN	98.79%
		CNN	98.65%
3	K.Kaharuddin and E. W. Sholeha [56]	KNN	77.50%
4	V. Allken et al. [55]	CNN	94 %
5	S. Villon et al. [52]	CNN	94.9%
	Proposed model		99.5%

The previous studies that have applied traditional machine learning and deep learning techniques to detect and classify fish species are presented in Table 4.4. We have been able to obtain a better classification performance. Our proposed model also ranks among the highest performance when compared with the rest of the related works.

4.10. Summary

In this chapter, we discussed the challenges encountered during the experiments, namely overfitting and oscillation in the training and validation accuracy or loss. These issues are attributed to the random sampling of the dataset. With each evaluation step, the dataset used is different, as different samples are taken, trained, and tested at each epoch. This variability in the dataset can lead to oscillation or overfitting. To address this, we used dropout layers into the model.

This chapter also provides details on the dataset used and the implementation of the proposed model for automatic fish species identification. The experimental evaluation of the model is described in depth, including the methodology and techniques employed. Furthermore, the test results are presented and compared with those of state-of-the-art models, specifically VGG16, Inceptionv3, and ResNet50. Finally, comparison of the proposed model with related works presented. Notably, the proposed model achieves higher classification accuracy compared to these existing models.

CHAPTER 5: CONCLUSION AND FUTURE WORK

5.1. Conclusion

This research work identifies convolutional neural network (CNN) as a significant and recent research area for automatic fish species identification using deep learning techniques. CNN possess local invariance properties that make them well-suited for detecting and classifying fish species automatically. However, after a thorough review of the literature and related works, it is evident that fish species identification and detection using deep learning techniques have received limited attention and remain unresolved problems.

In response to this gap, our research work makes contribution by developing a system that addresses the challenges faced in fish species identification. The system achieves better accuracy in automatic fish species identification using deep learning techniques, surpassing other pre-trained models such as VGG16, Inceptionv3, and ResNet50.

Specifically, our proposed model outperforms these pre-trained models in terms of training, testing accuracy, and loss. It achieves a training accuracy of 100% and a testing accuracy of 99.5%, demonstrating its ability to effectively learn and classify fish species. Compared to other models, our proposed model converges faster, resulting in lower loss values during training, making it more accurate and lightweight for deployment.

Our developed system, leveraging the CNN model, significantly advances fish species classification by improving accuracy in automatic identification. It addresses unresolved problems and reduces the reliance on expert skills. These findings contribute to the progress of accurate fish species identification, highlighting the significance of Deep learning technique, particularly CNN, in this field Progress of accurate fish species identification and hold promise for future developments in this field.

5.2. Contribution

The contribution of this study is listed below:

- Creation of a valuable dataset: A dataset was created from scratch, this contributing a valuable dataset which can be used for further research on endemic and native fish species in Ethiopia.
- State-of-the-art deep learning approach: The study proposed an advanced deep learning approach for identifying six fish species in Lake Hawassa, surpassing previous methods in accuracy.

- Contribution to CNN model for fish species in Ethiopia: The research contributed to the development of CNN model specifically designed for fish species in Ethiopia, improving the accuracy of identification.
- Comparative analysis with pre-trained models: The study demonstrated the superior performance of the proposed model compared to pre-trained models such as VGG16, Inceptionv3, and ResNet50, showcasing the effectiveness of the deep learning approach.

5.3. Future works

To further improve automatic fish species identification, several core future works can be explored.

- In this study we have considered only six fish species found in Hawassa Lake. We suggest expanding the dataset by including more different endemic fish species and increasing the number of images per species would enhance the model's accuracy and generalization capabilities.
- Transfer learning techniques can be employed by leveraging Different recent pre-trained models and fine-tuning them on the specific fish species dataset, potentially improving accuracy and reducing training time.
- Real-time implementation of the model in applications like underwater cameras or fish monitoring systems should be considered, optimizing the model for real-time performance. Enhancing the interpretability and explainability of the model's predictions would promote user understanding and trust.
- Integrating a hybrid approach that combines CNNs with other feature extraction techniques, exploring alternative feature extraction methods, utilizing ensemble methods, and addressing domain adaptation challenges would further enhance the accuracy, robustness, and adaptability of the fish species identification system.
- Additionally, improving the model's robustness to environmental variations, such as lighting conditions, water quality, and occlusions, is crucial. Collaboration with other researchers and organizations in the field, benchmarking results, and developing standardized datasets and evaluation metrics would foster advancements.

REFERENCES

[1] L. Ding and H. W. Kinnucan, "growing demand for animal-protein-source products in indonesia: trade implications," *J. Gender, Agric. Food Secur.*, vol. 1, no. 3, pp. 1–22, 2011.

[2] K. Kwasek, A. L. Thorne-Lyman, and M. Phillips, "Can human nutrition be improved through better fish feeding practices? a review paper," *Crit. Rev. Food Sci. Nutr.*, vol. 60, no. 22, pp. 3822–3835, 2020.

[3] S. Zhao *et al.*, "Application of machine learning in intelligent fish aquaculture: A review," *Aquaculture*, vol. 540, no. January, p. 736724, 2021.

[4] T. Oosting, B. Star, J. H. Barrett, M. Wellenreuther, P. A. Ritchie, and N. J. Rawlence, "Unlocking the potential of ancient fish DNA in the genomic era," no. December 2018, pp. 1513–1522, 2019.

[5] S. M. Lencha, J. Tränckner, and M. Dananto, "Assessing the water quality of lake hawassa Ethiopia—Trophic state and suitability for anthropogenic uses—applying common water quality indices," *Int. J. Environ. Res. Public Health*, vol. 18, no. 17, pp. 1–34, 2021.

[6] B. Tugrul, E. Elfatimi, and R. Eryigit, "Convolutional Neural Networks in Detection of Plant Leaf Diseases: A Review," *Agric.*, vol. 12, no. 8, 2022.

[7] M. Burić, L. Bavčević, S. Grgurić, F. Vresnik, J. Križan, and O. Antonić, "Modelling the environmental footprint of sea bream cage aquaculture in relation to spatial stocking design," *J. Environ. Manage.*, vol. 270, no. December 2019, 2020.

[8] H. Tayyab *et al.*, "Visual features based automated identification of fish species using deep convolutional neural networks," *Comput. Electron. Agric.*, no. September, p. 105075, 2019.

[9] Abebe Getahun, "The Freshwater Fishes of Ethiopia: Diversity and Utilization," *SINET Ethiop. J. Sci.*, vol. 21, no. 2, pp. 207–230, 2017.

[10] R. D. Ward, R. Hanner, and P. D. N. Hebert, "The campaign to DNA barcode all fishes, FISH-BOL," *J. Fish Biol.*, vol. 74, no. 2, pp. 329–356, 2009.

[11] A. Galimberti *et al.*, "DNA barcoding as a new tool for food traceability," *Food Res. Int.*, vol. 50, no. 1, pp. 55–63, 2013.

[12] Mengesha and T. Awoke, "Fish Species Diversity in Major River Basins of Ethiopia : A Review," *World J. Fish Mar. Sci.*, vol. 7, no. 5, pp. 365–374, 2015.

[13] K. Peffers, T. Tuunanen, M. A. Rothenberger, and S. Chatterjee, “A design science research methodology for information systems research,” *J. Manag. Inf. Syst.*, vol. 24, no. 3, pp. 45–77, 2007.

[14] M. Grandini, E. Bagli, and G. Visani, “Metrics for Multi-Class Classification: an Overview,” pp. 1–17, 2020.

[15] A. Kebede, T. Meko, A. Hussein, and Y. Tamiru, “Review on Opportunities and Constraints of Fishery in Ethiopia,” *Int. J. Poult. Fish. Sci.*, vol. 1, no. 1, pp. 1–4, 2017.

[16] C. Béné and S. Heck, “Fish and Food Security in Africa,” *Aqua Docs*, vol. 28, no. 3, pp. 5–12, 2005.

[17] M. Wakjira, “Fisheries and aquaculture,” *Routledge Handb. Food Nutr. Secur.*, no. June 2011, pp. 153–168, 2016.

[18] A. Tilahun and A. Engdawork, “Isolation, Identification and Antimicrobial Susceptibility Profile of E.Coli (O157: H7) from Fish in Lake Hawassa, Southern Ethiopia,” vol. 47, no. 4, pp. 124–134, 2021.

[19] R. C. Gonzalez and R. E. Woods, *4th edition Digital image processing*. 2018.

[20] V. P. Reddy, “lecture notes on digital image processing iv b.tech i semester (jntuh-r15) institute of aeronautical engineering (autonomous) dundigal, hyderabad-500043”.

[21] K. Saarinen, *Image processing, analysis and machine vision*, vol. 35, no. 1. 1994.

[22] A. H. Lone and A. N. Siddiqui, “Noise models in digital image processing,” *Glob. Sci-Tech*, vol. 10, no. 2, p. 63, 2018.

[23] William K. Pratt, *Digital image processing fourth edition*, vol. 4, no. 1. 2007.

[24] W. S. Noble, “What is a support vector machine?,” *Nat. Biotechnol.*, vol. 24, no. 12, pp. 1565–1567, 2006.

[25] A. Enquhone, “Ethiopian maize diseases recognition and classification using support vector machine,” *Int. J. Comput. Vis. Robot.*, vol. 9, no. 1, pp. 90–109, 2019.

[26] E. Keogh, “Naïve Bayes Classifier,” 2006.

[27] S. R. Bhagya Shree and H. S. Sheshadri, “Diagnosis of Alzheimer’s disease using Naive Bayesian Classifier,” *Neural Comput. Appl.*, vol. 29, no. 1, pp. 123–132, 2018.

[28] Suparyanto dan Rosad (2015, “Deep learning,” vol. 5, no. 3, pp. 248–253, 2020.

[29] C. R. Alimboyong, A. A. Hernandez, and R. P. Medina, “Classification of Plant Seedling Images Using Deep Learning,” *IEEE Reg. 10 Annu. Int. Conf. Proceedings/TENCON*, vol. 2018-Octob, no. October, pp. 1839–1844, 2019.

[30] H. Endrias, “Plant Disease Detection and Classification Using Artificial Neural Network,” *Addis Ababa Univ. Repos.*, vol. 01, p. 83, 2019.

[31] M. Ranjan, M. Rajiv Weginwar, N. Joshi, and A. Ingole, “Detection and classification of leaf disease using artificial neural network,” *Int. J. Tech. Res. Appl.*, vol. 3, no. 3, pp. 331–333, 2015.

[32] J. Brownlee, “Master Machine Learning Algorithms: Discover how they work and implement them from scratch.,” *Mach. Learn. Mastery*, pp. 1–163, 2016.

[33] M. I. Razzak, S. Naz, and A. Zaib, “Deep learning for medical image processing: Overview, challenges and the future,” *Lect. Notes Comput. Vis. Biomech.*, vol. 26, pp. 323–350, 2018.

[34] T. Bezdan and N. Bačanin Džakula, “Convolutional Neural Network Layers and Architectures,” no. July, pp. 445–451, 2019.

[35] Y. Wang, Y. Li, Y. Song, and X. Rong, “The influence of the activation function in a convolution neural network model of facial expression recognition,” *Appl. Sci.*, vol. 10, no. 5, 2020.

[36] Z. Teng, S. Teng, J. Zhang, G. Chen, and F. Cui, “Structural damage detection based on real-time vibration signal and convolutional neural network,” *Appl. Sci.*, vol. 10, no. 14, pp. 1–16, 2020.

[37] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: A simple way to prevent neural networks from overfitting,” *J. Mach. Learn. Res.*, vol. 15, pp. 1929–1958, 2014.

[38] Shefraw and A. Asfaw, “Deep Learning Approach for Ethiopian Banknote Denomination Classification and Fake Detection System,” *Comput. Sci. Control Eng.*, vol. 7, no. 4, pp. 30–37, 2020.

[39] Linigerew Mengstie Shita, “Ethiopian Coffee Leaf Diseases Identification Using Deep Learning,” *J. Emerg. Technol. Innov. Res.*, vol. 8, no. 10, pp. 498–508, 2021.

[40] B. A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet Classification with Deep Convolutional Neural Networks,” *Commun. ACM*, vol. 60, no. 6, pp. 84–90, 2012.

[41] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *3rd Int. Conf. Learn. Represent. ICLR 2015 - Conf. Track Proc.*, pp. 1–14, 2015.

[42] C. Szegedy *et al.*, “Going deeper with convolutions,” *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 07-12-June, no. January 2017, pp. 1–9, 2015.

[43] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 2016-Decem, pp. 770–778, 2016.

[44] D. Rathi, S. Jain, and S. Indu, “Underwater Fish Species Classification using Convolutional Neural Network and Deep Learning,” *2017 9th Int. Conf. Adv. Pattern Recognition, ICAPR 2017*, pp. 344–349, 2018.

[45] D. Bekkozhayeva and P. Cisar, “Image-Based Automatic Individual Identification of Fish without Obvious Patterns on the Body (Scale Pattern),” *Appl. Sci.*, vol. 12, no. 11, 2022.

[46] hernandes alexander; jesmar Francis, “Classification of Fish Species with Augmented Data using Deep Convolutional Neural Network,” *IEEE*, vol. 6, 2019.

[47] H. Malik, A. Naeem, S. Hassan, F. Ali, R. A. Naqvi, and D. K. Yon, *Multi-classification deep neural networks for identification of fish species using camera captured images*, vol. 18, no. 4 April. 2023.

[48] P. Hridayami, I. K. G. D. Putra, and K. S. Wibawa, “Fish species recognition using VGG16 deep convolutional neural network,” *J. Comput. Sci. Eng.*, vol. 13, no. 3, pp. 124–130, 2019.

[49] F. Kratzert and H. Mader, “Fish species classification in underwater video monitoring using Convolutional Neural Networks,” no. May, 2018.

[50] S. N. M. Rum and F. A. Z. Nawawi, “FishDeTec: A Fish Identification Application using Image Recognition Approach,” *Int. J. Adv. Comput. Sci. Appl.*, vol. 12, no. 3, pp. 102–106, 2021.

[51] M. A. Iqbal, Z. Wang, Z. A. Ali, and S. Riaz, “Automatic Fish Species Classification Using Deep Convolutional Neural Networks,” *Wirel. Pers. Commun.*, vol. 116, no. 2, pp. 1043–1053, 2021.

[52] S. Villon *et al.*, “A Deep learning method for accurate and fast identification of coral reef fishes in underwater images,” *Ecol. Inform.*, vol. 48, no. August, pp. 238–244, 2018.

[53] B. V. Deep and R. Dash, “Underwater Fish Species Recognition Using Deep Learning Techniques,” *2019 6th Int. Conf. Signal Process. Integr. Networks, SPIN 2019*, pp. 665–669, 2019.

[54] I. M. Yusup, M. Iqbal, and I. Jaya, “Real-time reef fishes identification using deep learning,” *IOP Conf. Ser. Earth Environ. Sci.*, vol. 429, no. 1, 2020.

[55] V. Allken, N. O. Handegard, S. Rosen, T. Schreyeck, T. Mahiout, and K. Malde, “Fish species identification using a convolutional neural network trained on synthetic data,” *ICES J. Mar. Sci.*, vol. 76, no. 1, pp. 342–349, 2019.

[56] K. Kaharuddin and E. W. Sholeha, “Classification of Fish Species with Image Data Using K-Nearest Neighbor,” *Int. J. Comput. Inf. Syst.*, vol. 2, no. 2, pp. 54–58, 2021.

[57] K. K. Dobbin and R. M. Simon, “Optimally splitting cases for training and testing high dimensional classifiers,” *BMC Med. Genomics*, vol. 4, no. 1, p. 31, 2011.

APPENDIX A

Sample code for import the images from the dataset path and apply rescaling

```
train_dir = '/content/drive/MyDrive/6000_70_30/train'
val_dir = '/content/drive/MyDrive/6000_70_30/val'
test_dir = '/content/drive/MyDrive/6000_70_30/test'

#Import important libraries
from tensorflow.keras.preprocessing.image importImageDataGenerator

train_datagen = ImageDataGenerator(rescale=1./255)
val_datagen = ImageDataGenerator(rescale=1./255)
test_datagen = ImageDataGenerator(rescale=1./255)

train_generator = train_datagen.flow_from_directory(
    train_dir,
    target_size=(224, 224),
    batch_size=64,
    class_mode='categorical'
    )
val_generator = val_datagen.flow_from_directory(
    val_dir,
    target_size=(224, 224),
    batch_size=64,
    shuffle = False,
    class_mode='categorical'
)
test_generator = test_datagen.flow_from_directory(
    test_dir,
    target_size=(224, 224)
    )
```

APPENDIX B

Sample Code for Model Design

```
# Import important libraries

from tensorflow.keras.models import Sequential, Model
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout

input_shape= (224, 224, 3)
n_classes = 6

emu_model = Sequential()
emu_model.add(Conv2D(16,kernel_size =(3,3),activation='relu',
strides=1, padding = 'Same', input_shape=input_shape))
emu_model.add(Conv2D(16,kernel_size =(3,3), activation='relu',strides=1,padding =
'Same'))
emu_model.add(MaxPooling2D((2, 2)))
emu_model.add(Conv2D(32,  kernel_size = (3,3), activation='relu',strides=1,padding =
'Same'))
emu_model.add(Conv2D(32,  kernel_size = (3,3), activation='relu',strides=1,padding =
'Same'))
emu_model.add(MaxPooling2D((2, 2)))
emu_model.add(Conv2D(64,  kernel_size = (3,3), activation='relu',strides=1,padding =
'Same'))
emu_model.add(Conv2D(64,  kernel_size = (3,3), activation='relu',strides=1,padding =
'Same'))
emu_model.add(MaxPooling2D((2, 2)))
emu_model.add(Dropout((0.25)))
emu_model.add(Conv2D(128,  kernel_size = (3,3), activation='relu',strides=1,padding =
'Same'))
emu_model.add(Conv2D(256,  kernel_size = (3,3), activation='relu',strides=1,padding =
'Same'))
emu_model.add(MaxPooling2D((2, 2)))
emu_model.add(Dropout((0.25)))
emu_model.add(Flatten())
emu_model.add(Dense(512, activation='relu'))
emu_model.add(Dropout((0.4)))
emu_model.add(Dense(n_classes, activation='softmax'))
```

APPENDIX C

Sample code for image resizing and median filtering

```
#import important libraries

import glob
import cv2
images_path = glob.glob('/content/drive/MyDrive/catfish/*.jpg')
i = 0
for image in images_path:
    img = cv2.imread(image)
    blur = cv2.medianBlur(img, 5)
    img_resized = cv2.resize(img, (224,224))
    cv2.imwrite('/content/drive/MyDrive/catfishresize/catfish%12i.jpg' %i, img_resized)
    i +=1
```