Multi-label Coffee Bean Classification Using Deep Learning

By

Chernet Ewawey

ADVISOR: Michael Melese (PhD)

A THESIS SUBMITTED TO

HAWASSA UNIVERSITY, INSTITUTE OF TECHNOLOGY,

DEPARTMENT OF COMPUTER SCIENCE,

SCHOOL OF GRADUATE STUDIE

HAWASSA, ETHIOPIA

October, 2024

Hawassa, Ethiopia

INSTITUTE OF TECHNOLOGY

FACULTY OF INFORMATICS

MSC PROGRAM IN COMPUTER SCIENCE

Multi-label Coffee Bean Classification Using Deep Learning

A Thesis Submitted to Hawassa University, Institute of Technology, Faculty of Informatics, Department of Computer science in Partial Fulfillment of the Requirements for the Degree of Master of Science in Computer Science

By: Chernet Ewawey

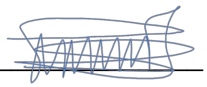
Advisor: Dr. Michael Melese

October 2024

Hawassa, Ethiopia

APPROVAL SHEET-I

This is to certify that the thesis entitled, "**Multi -label coffee bean classification using deep learning**" submitted in partial fulfillment of the requirements for the degree of Master's with specialization in Computer Science, the Graduate Program of the Faculty of Informatics, and has been carried out by **Chernet Ewawey Weldemikael.** Therefore, we recommend that the student has fulfilled the requirements and hence hereby can submit the thesis to the department.

Michael Melese (PhD)		30/11/2024
Name of major advisor	Signature	Date

Name of co-advisor	Signature	Date

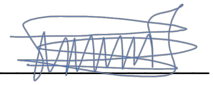
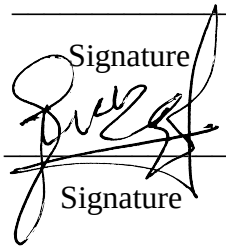
EXAMINERS' APPROVAL SHEET

SCHOOL OF GRADUATE STUDIES

HAWASSA UNIVERSITY EXAMINERS' APPROVAL SHEET

(Submission Sheet-2)

We, the undersigned, members of the Board of Examiners of the final open defense by **Chernet Ewawey Weldemikael** have read and evaluated his/her thesis entitled "**Multi -label coffee bean classification using deep learning**", and examined the candidate. This is, therefore, to certify that the thesis has been accepted in partial fulfillment of the requirements for the degree.

Michael Melese (PhD)		30/11/2024
Name of Major Advisor	Signature	Date
Andargachew Mekonnen (PhD)		Nov. 29, 2024
Name of Internal Examiner-I	Signature	Date
Name of Internal Examiner-II	Signature	Date
Siraj Sebhatu(Ph.D)		29/11/2024
Name of External examiner	Signature	Date
SGS Approval	Signature	Date

Final approval and acceptance of the thesis is contingent upon the submission of the final copy of the thesis to the School of Graduate Studies (SGS) through the Department/School Graduate Committee (DGC/SGC) of the candidate's department.

Stamp of SGS Date: _____

Declaration of Authorship

I, Chernet Ewawey, declare that this thesis work titled “Multi-label Coffee Bean Classification Using Deep Learning” along with the experiment and the result presented in it is my original work. This work is not submitted in whole or in part for any publication and specific integration and reference are made to the work of others. Other sources are acknowledged by citation by giving an explicit reference.

I have undertaken the study independently with the guidance and support of my research advisor.

Date: 2/12/2024

Signature 

Chernet Ewawey

Acknowledgment

First and foremost, praises and thanks to God, the Almighty, for His showers of blessings throughout my research work, enabling me to complete the research successfully. I would like to express my deep and sincere gratitude to my research advisor, Dr. Michael Melese, from the School of Information Science at Addis Ababa University, for providing invaluable guidance throughout this research. His dynamism, vision, sincerity, and motivation have deeply inspired me. He has taught me the methodology to carry out the research and present the research work as clearly as possible. It was a great privilege and honor to work and study under his guidance. I am extremely grateful for what he has offered me. I would also like to thank him for his friendship, empathy, and great sense of humor.

I am extremely grateful to my parents for their love, prayers, caring, and sacrifices in educating and preparing me for my future. I'm also very much thankful to my friends and colleagues for supporting and motivating me to complete this thesis successfully. They were there for me when I needed them. A warm thanks to the manager and workers of ECQIC for helping and providing me with coffee bean samples for this work. They showed me their deepest gratitude and assisted me in collecting coffee bean samples from each coffee class.

Most importantly, I would like to thank my wife, Kibre Abera, for her unwavering support and encouragement throughout all the challenges. Finally, my thanks go to all the people who have supported me in completing the research work, directly or indirectly.

.

Abbreviations

AI	Artificial Intelligence
ANN	Artificial Neural Network
API	Application Program Interface
CNN	Convolutional Neural Network
Conv	Convolutional
DIP	Digital Image Processing
DL	Deep Learning
DP	Dropout
DS	Design Science
DSR	Design Science Research
DSRP	Design Science Research Process
ECQIC	Ethiopian Coffee Quality and Inspection Center
ECX	Ethiopian Commodity Exchange
FC	Fully Connected
GPU	Graphical Processing Unit
Keras	Key Real-time Artificial Intelligence
ML	Machine Learning
ReLU	Rectified Linear Unit
TensorFlow	Open-source Machine Learning Library

Table of Contents

List of Figures

List of Tables

Abstract

Coffee is a critical agricultural product and a significant economic commodity, particularly in Ethiopia, where it serves as a major export. Traditional methods of coffee bean assessment are labor-intensive and prone to human error, necessitating the development of automated, accurate solutions. This study addresses this problem by applying deep learning techniques, specifically Convolutional Neural Networks (CNNs), to classify coffee beans based on their quality and geographical origin. The primary objective of this research is to design and develop a robust CNN model capable of accurately classifying coffee beans using image data. To achieve this, we collected an extensive dataset of 3,965 coffee beans from the Ethiopian regions of Arsi, Yirgacheffe, Guji, and Sidama, photographing each bean from the front and back to obtain a total of 6,373 images. To enhance the dataset's balance, an additional 2,417 images were generated through augmentation, bringing the final dataset to 8,790 images. The methodology involved preprocessing these images, followed by training and evaluating multiple CNN architectures. Techniques employed include the use of both grayscale and RGB image data, Dropout layers to prevent over-fitting, and the Adam optimizer for efficient training. Our results indicate that the CNN model trained on RGB images achieved a peak accuracy of 99.66% in a specific experiment, while the average accuracy across all experiments was 99.06%. This highlights the significance of color information in enhancing feature extraction and model learning. The model's high accuracy and efficiency demonstrate its potential to automate and improve coffee quality assessment, offering a valuable tool for the coffee industry. By leveraging deep learning techniques, this study provides a scalable and precise solution for the classification of coffee beans, potentially transforming quality control processes and setting new standards in the coffee industry. The study's findings underscore the superiority of CNN-based models in handling complex image data, particularly when color features play a crucial role in classification tasks.

Keywords *Deep Learning, Coffee Bean Classification, Digital Image Processing, Ethiopian Coffee, Multi-label Classification, Quality Control, Machine Learning, Data Preprocessing, Model Evaluation*

CHAPTER ONE
INTRODUCTION

1.1 Background

1.1.1 Agricultural Background

Agriculture is the cornerstone of the global economy, providing essential resources and livelihoods for billions of people. It plays a critical role in food security, economic development, and environmental sustainability. In many developing countries, agriculture is the primary source of income and employment, making it a vital sector for national development. The advancement of agricultural practices and technologies has significantly increased productivity and efficiency, allowing for the cultivation of a wide range of crops that support the global food supply.

1.1.2 Coffee Background

Among these crops, coffee stands out as one of the most consumed beverages in the world, representing the second-largest exporting commodity [1]. Coffee trees, which can grow up to more than 30 feet (9 meters) in height [2], are typically pruned short to conserve energy and facilitate harvesting. The trees are characterized by pairs of lush, glossy leaves that grow opposite each other along the branches. Coffee cherries, the fruit from which coffee beans are harvested, grow along these branches. Due to the continuous growth cycle of the coffee tree, it is common to find flowers, green fruit, and ripe fruit simultaneously on a single tree.

Coffee is an edible commodity, widely consumed as a beverage, and increasingly used as an input in food processing industries [3]. It is a seasonal crop, with seasons varying from country to country, starting and finishing at different times throughout the year [4]. Coffee is the backbone of the Ethiopian economy, leading to generating foreign exchange for the country [5]. The commercial coffee industry primarily focuses on two important species Arabica and Robusta [1], [3], [4], [8]. The origin of Coffee Arabica has been traced to Ethiopia, while Robusta Coffee was believed to come from Central to West Africa [9]. Arabica Coffee is the most widely consumed, dominating over 70% in volume of production and over 90% of traded value globally [5].

Ethiopia is the native home of the Arabica coffee species, where the coffee plant first originated. Today, Arabica coffee is cultivated in various regions across the globe. Within Ethiopia, diverse

coffee varieties thrive in distinct regions. Ethiopian coffee beans can be categorized as Yirgacheffe, Jimma, Harrar, Bebeka, Teppi, Limmu, Sidamo, Lekempti, Illubabor, Welega, Arsi, Guji, Bale, and Gimbi is based on their growing areas [10], [11], [12].

The quality of coffee is of vital importance to the coffee industry [13]. High-quality coffee is characterized by a clean and appealing appearance in both raw and roasted forms, a pleasant aroma, and a satisfying cup taste. Numerous factors influence coffee quality [14]. The geographical origin, soil characteristics, agricultural practices, climatic conditions, harvesting methods, and post-harvest processing techniques are the major attributes of the deterioration of coffee quality [15]. Issues with coffee quality are primarily linked to inadequate postharvest processing and handling practices. These include drying on bare ground, improper wet processing, suboptimal storage and transportation, as well as poor agronomic practices, such as uncontrolled shade levels, insufficient pruning and weeding. Additionally, poor harvesting methods, such as stripping and collecting fallen fruits from the ground, contribute significantly to quality problems. 90% of Ethiopian coffee is hand-picked in forests or home gardens, and just 10% of it comes from organized farming [11], [12].

Trade and export are done in various ways [5]. Nowadays, they are controlled entirely by the national authorities, and all coffee is evaluated and sold through an auction system in Addis Ababa the Ethiopian Commodity Exchange (ECX) [12]. Ethiopia is renowned for its rich coffee heritage, and the coffee industry plays a significant role in the country's economy. Coffee bean classification is vital for ensuring quality and maintaining Ethiopia's global reputation in the coffee market. Traditional coffee bean classification is labor-intensive and subjective, often leading to inconsistencies. The research aims to implement deep learning techniques to develop a multi-label label coffee bean classification system that can improve the efficiency and accuracy of coffee origin and grading in the Ethiopian context.

1.1.3 AI and Machine Learning in Agriculture

Artificial Intelligence (AI) and Machine Learning (ML) have increasingly become integral to modern agricultural practices, offering innovative solutions to longstanding challenges. AI and ML technologies are being leveraged to optimize crop yields, enhance precision farming, and improve supply chain efficiency. In the context of coffee, these technologies are particularly valuable for automating labor-intensive processes such as quality assessment and classification.

Machine Learning (ML), a branch of Artificial Intelligence (AI), focuses on creating algorithms that allow computers to learn from data and make data-driven decisions. Deep Learning (DL), an advanced subset of ML, utilizes multi-layered neural networks to capture and model intricate patterns within large datasets. Convolutional Neural Networks (CNNs), a type of DL algorithm, have proven particularly effective in image recognition tasks, making them ideal for applications such as coffee bean classification, where visual features are crucial.

1.1.4 Integrating Agriculture, Coffee, and AI/ML

The integration of AI and ML with agricultural practices, particularly in the coffee industry, represents a significant advancement in both fields. By applying DL techniques like CNNs to the classification of coffee beans, this research aims to address the challenges associated with traditional, manual methods. The goal is to design and develop a multi-label coffee bean classification system that improves the efficiency and accuracy of assessing coffee origin and quality in Ethiopia.

This research will focus on four key coffee-growing regions in Ethiopia, leveraging image data to train and evaluate CNN models. The ultimate objective is to create a scalable and precise solution for the classification of Ethiopian coffee beans, ensuring consistent quality and enhancing Ethiopia's global reputation in the coffee market.

1.2 Motivation of the study

The study Multi-label Coffee Bean Classification using Deep Learning is motivated by the need to address significant challenges faced by coffee producers, particularly in regions like the Gedeo Zone, known for its high-quality coffee farms. The current methods of coffee bean classification are largely manual, time-consuming, and prone to human error, which can lead to inconsistencies in quality assessment. In a competitive global market, where the reputation of coffee is closely tied to its quality and origin, there is a pressing need for more accurate and efficient classification techniques.

The motivation for this research stems from the potential to leverage deep learning technologies to improve the precision and reliability of coffee bean classification. By developing a robust multi-label classification system, this study aims to provide a scalable solution that can enhance the decision-making process for coffee producers. This includes enabling more accurate identification

of coffee bean quality and variety, leading to better resource allocation, higher yields, and improved market positioning.

Furthermore, the research addresses the broader objective of supporting the Ethiopian coffee industry in maintaining and elevating its global reputation. As Ethiopia is the birthplace of Arabica coffee, ensuring that its coffee meets international quality standards is crucial for sustaining the country's economic growth and securing its position in the global coffee market.

This study is driven by the potential to make a meaningful impact on the coffee industry by providing a practical solution to a real-world problem, ultimately contributing to the economic and social well-being of coffee-producing communities.

1.3 Statement of the Problem

In the agricultural industry, ensuring the quality of various products, such as grain seeds, remains a significant challenge. Traditionally, the quality and variety of these seeds have been assessed manually through visual inspection by experienced technicians. This method, while widely used, is labor-intensive, time-consuming, and susceptible to human error, leading to inconsistent results and variability in quality [16].

Coffee a crucial commodity for Ethiopia represents a significant portion of the country's economy and exports. The Ethiopian coffee industry is marked by a complex production process that involves multiple stakeholders, including farmers, processors, and traders [17]. Despite its importance, the industry faces persistent challenges that affect the quality and consistency of coffee beans. One of the key issues is the lack of an efficient and accurate classification system for coffee beans.

Traditional coffee bean classification methods rely heavily on manual sorting and grading, which are not only time-consuming but also prone to errors. The reliance on human judgment can lead to inconsistencies in quality assessment, which can negatively impact the reputation of Ethiopian coffee on the global market [3]. The manual nature of this process makes it difficult to achieve uniform standards, especially when dealing with large volumes of coffee beans.

The need for a more reliable and efficient classification system is crucial to ensure that Ethiopian coffee maintains its high standards and meets international expectations. By reducing human error and speeding up the process, a more advanced system could enhance the quality control process.

The Previous research has focused on developing individual classification models that consider either the origin or the grade of coffee beans [16], [17], [18], [3], [15], [19]. However, these models have limitations in capturing the complex interactions between various characteristics of coffee beans, such as color, shape, size, and texture. These oversights can lead to inaccurate classifications, affecting the consistency and overall quality of the coffee beans.

Even with the use of machine learning and deep learning algorithms, previous approaches have struggled to predict the quality and consistency of coffee beans accurately based on their origin and grade. This limitation is particularly problematic in the Ethiopian context, where the diversity of coffee beans and their growing conditions make classification more complex. Misclassifications can lead to misunderstandings and disputes between buyers and sellers, undermining confidence in Ethiopian coffee.

To address this issue, there is a need for a multi-label classification system that can simultaneously evaluate multiple aspects of coffee beans. By improving the accuracy of these predictions, the system can help to ensure that Ethiopian coffee consistently meets quality standards, thereby strengthening its position in the global market.

This research aims to design and develop a deep learning-based multi-label classification system that combines both origin and grade features to classify coffee beans. The system will be trained on a dataset of coffee beans from four different coffee-growing regions in Ethiopia. By addressing the limitations of traditional methods and previous models, this research seeks to provide a more accurate and reliable classification system that enhances the quality and consistency of Ethiopian coffee in the global market.

1.4 Research Questions

1. Which image features grayscale or RGB are most effective in categorizing the quality and geographical origin of Ethiopian coffee beans using deep learning techniques?
2. To what extent does the CNN model accurately identify the source and quality of Ethiopian coffee beans when applied to a multi-label classification task?

1.5 General Objective

The general objective of this research is to design and implement a deep learning-based system for the multi-label coffee bean classification.

1.5.1 Specific Objectives

These specific objectives outline the step-by-step process of the research, from data collection and model development to evaluation and recommendations.

- Conduct a comprehensive literature review
- Collect, preprocess, and standardize a diverse and representative dataset of Ethiopian coffee bean images.
- Design and implement a Convolutional Neural Network (CNN) model capable of accurately classifying coffee beans by quality and geographical origin.
- Train the CNN model on the processed dataset to learn features necessary for effective classification.
- Evaluate the performance of the model using appropriate metrics to assess its accuracy and effectiveness.
- Report the findings of the study and provide recommendations based on the results.

1.6 Methodology

Methodology is the strategy or plan of action that outlines the procedures, techniques, and tools used to collect and analyze data. A well-designed methodology is crucial for ensuring the reliability and validity of research findings.

1.6.1 Literature Review

A literature review is a critical and comprehensive analysis of existing literature on a specific topic or research question. It is a crucial component of academic research and aims to provide an overview of relevant studies, theories, concepts, and findings related to the chosen subject.

1.6.2 Research Design

This study follows an experimental research design aimed at developing and evaluating a deep-learning model for multi-label coffee bean classification. The design involves several key steps, including data collection, preprocessing, model design, training, evaluation, and reporting. The experimental nature of the research allows for testing different variables, such as image features (grayscale vs. RGB) and hyperparameters, to determine their impact on the model's performance.

1.6.3 Data Collection

In this study, the dataset employed throughout the entire process comprises digital images of coffee beans. These coffee beans are sourced from the Ethiopian Coffee Quality and Inspection Center (ECQIC) found in the Hawassa Branch. The collection encompasses coffee beans cultivated in four distinct regions of Ethiopia. focus on coffee bean image classification, each coffee bean is photographed using a digital camera. These images of coffee beans serve as the training, validation, and testing data for the model.

1.6.4 Data Preprocessing

Data preprocessing involved cleaning and standardizing the collected images to prepare them for analysis. This step included resizing images, normalizing pixel values, and applying augmentation techniques such as rotation, flipping, and scaling to enhance the robustness of the model. The preprocessing also involved converting images to grayscale and RGB formats to compare the effectiveness of these features in the classification process.

1.6.5 Model Design

The research designed a Convolutional Neural Network (CNN) architecture tailored for the multi-label classification of coffee beans. The model's architecture was selected based on insights gained from the literature review, incorporating multiple layers of convolution, pooling, and dropout to prevent over-fitting. The model's hyperparameters, such as learning rate, batch size, and the number of epochs, were tuned to optimize performance.

1.6.6 Feature Extraction

Feature extraction focuses on identifying the most relevant characteristics of coffee beans, such as color, texture, and shape. The CNN model automatically learned these features from the images during training, enabling the model to distinguish between different quality grades and origins accurately.

1.6.7 Model Training

The CNN model was trained on the preprocessed dataset using the Adam optimizer, which was selected for its efficiency in training deep learning models. The training process involved multiple iterations to fine-tune the model and improve its accuracy.

1.6.8 Evaluation Technique

Once the solution reaches an adequate stage, assessment can commence on the design artifact. Testing the designed artifact or assessing how effectively the developed model supports a solution to a problem involves utilizing the test dataset with the classifier constructed from the training dataset. Evaluating the model's performance entails comparing its output with observed data using performance measurement metrics and percentage accuracy measures for each class, such as recall, f1-score, and precision values, to evaluate diagnostic accuracy. The confusion matrix is commonly used for binary classification, where one class is labeled as the positive class and the other as the negative class [2]. Within this framework, the matrix consists of four components: true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN) [3].

1.6.9 Reporting and Analysis

Finally, the research findings were reported, focusing on the model's performance and the impact of different features and hyperparameters. The results were examined to highlight the model's strengths and limitations, offering recommendations for future research in the area of coffee bean classification.

1.7 Scope of Study and Limitation

This research focuses on applying deep learning techniques to classify Ethiopian coffee beans based on their physical properties, including attributes such as shape, color, and size, as well as their regional origin and quality levels. The study involves designing a computer vision-based model using Convolutional Neural Networks (CNNs) to analyze images of coffee beans. Specifically, the research targets beans from four Ethiopian regions: Arsi, Yirgacheffe, Guji, and Sidama.

The study has several limitations. Firstly, it does not include the classification of different coffee types (e.g., Arabica vs. Robusta) or the analysis of chemical properties such as caffeine content or acidity. The focus is strictly on visual characteristics and regional attributes, excluding any chemical or typological factors.

Additionally, the research is confined to coffee beans from four specific Ethiopian regions. Consequently, the findings may not apply to beans from other Ethiopian regions or coffee-

producing countries outside of Ethiopia. The model's performance is evaluated solely based on beans from Arsi, Yirgacheffe, Guji, and Sidama, limiting its generalizability to other areas.

The study also does not account for variations in environmental conditions or post-harvest processing methods that could affect coffee bean quality. Factors such as soil type, climate, and processing techniques are not considered in the model, which may impact the broader applicability of the results. The dataset used for training and evaluation consists only of images from the specified Ethiopian regions. This research classifies the location and quality of Ethiopian coffee.

1.8 Significance of Study

This study has the potential to benefit Ethiopian coffee producers, traders, and consumers by improving coffee quality assessment and enhancing the value of coffee products. It can contribute to the country's coffee industry, making it more competitive in the global market. Implementing deep learning for coffee bean classification enhances the accuracy and consistency of grading. This, in turn, ensures that only high-quality coffee beans meet international standards, improving the reputation of Ethiopian coffee in the global market. The study's significance lies in its potential to enhance the Ethiopian coffee industry's quality, economic sustainability, and global competitiveness. It represents a significant step towards modernizing and optimizing coffee bean classification, benefiting coffee producers, processors, and the Ethiopian economy as a whole. Traditional manual grading is labor-intensive and subject to human error. So automating the process using deep learning can save time and reduce the costs associated with manual labor in coffee processing.

1.9 Organization of the Thesis

The study is structured into five chapters.

The **initial Chapter** introduces the study's objectives, statement of problem, research methodology, significance, scope, and limitations. It also examines the research design from the perspective of design science research methodology.

Chapter Two comprises a review of the literature and related works. It introduces Ethiopian coffee varieties, digital image processing, applications of image processing, image classification techniques, and deep learning. Additionally, it discusses international and local related works and their accomplishments.

The third Chapter focuses on system design and modeling, providing a detailed explanation of the design and development of coffee bean classification. This includes the general system architecture, data preparation, and pre-processing, as well as the architecture and components of the model.

Chapter Four Discusses the demonstration and evaluation of the model, providing detailed insights into how the model is trained and tested in each experiment. Evaluation results are presented at the end of each experiment, with comparisons of different experiment results leading to the selection of the best-performing model. This chapter also covers the design and development of a prototype for end-users.

Finally, **Chapter Five** discusses the conclusion, recommendations, and future work for further enhancing the study and its practical applications.

CHAPTER TWO

Literature Review and Related Works

2.1 Overview

In this chapter, both conceptual and related literature reviews are described. The conceptual review focuses on the agricultural industry, coffee usage, and the significance of coffee exports, particularly in the Ethiopian context. Additionally, the review discusses the deep learning algorithm CNN, and performance evaluation metrics relevant to coffee bean classification. In the related literature section, articles and research work related to the study are critically reviewed and summarized to highlight gaps in the existing research.

2.2 Conceptual Discussion

2.2.1 Agriculture and Its Global Importance

Agriculture is a fundamental component of the global economy, providing food, raw materials, and employment to billions of people worldwide [4]. It plays a crucial role in sustaining livelihoods, ensuring food security, and driving economic growth, especially in developing countries where it is often the primary source of income [5]. Over the years, advancements in agricultural practices and technologies have led to increased productivity and efficiency, contributing to the diversification of crops and supporting the global food supply chain. However, despite these advancements, the agricultural sector still faces challenges such as climate change, soil degradation, and the need for sustainable farming practices [6].

Agriculture plays a crucial role in promoting economic equity and fostering global prosperity [5]. For instance, since 2000, Sub-Saharan Africa has experienced an agricultural growth rate of approximately 4.3% annually, surpassing that of any other region and contributing significantly to its economic progress, as reported by the United States Agency for International Development (USAID). However, globally, agricultural employment has declined, with jobs decreasing from 1 billion in 2000 to 883 million in 2019 [6].

2.2.2 Coffee Usage

Coffee ranks among the most popular beverages worldwide, cherished for its stimulating effects and rich flavor[7]. The importance of coffee in culture varies across regions, but it is often

associated with social gatherings, hospitality, and daily rituals. Beyond its traditional use as a beverage, coffee has become integrated into various culinary applications, including desserts, sauces, and even cosmetics. The versatility of coffee extends to its role in health and wellness, with studies suggesting potential benefits such as improved cognitive function and reduced risk of certain diseases [8]. However, excessive consumption has also been linked to negative health effects, making moderation key.

2.2.3 Coffee Export and Its Economic Status

Coffee is a major agricultural export, with the global coffee market generating billions of dollars annually [9]. Countries like Brazil, Vietnam, and Ethiopia are among the top coffee producers and exporters [7]. For Ethiopia, coffee is not just a crop but a vital part of the national economy, providing livelihoods for millions of people [8]. The country's unique coffee varieties are highly prized commodities on the international market, contributing significantly to foreign exchange earnings. Despite its importance, the Ethiopian coffee industry has to deal with fluctuating market prices, climate change, and the need for improved infrastructure and technology in the production and processing stages [7].

2.3 Ethiopian Coffee

2.3.1 Historical and Cultural Significance

Coffee is believed to have been discovered in the 9th century by a young goat herder named Kaldi from the Kaffa region [21]. Kaldi noticed his goats behaving energetically and playfully after consuming red berries from an unfamiliar shrub. Curious, he decided to try the berries himself and experienced an invigorating surge of energy [7]. Enthusiastic about his discovery, Kaldi shared it with local monks. Although initially skeptical, the monks eventually soaked the berries in hot water and found the resulting beverage not only delicious but also effective in helping them stay alert during long prayer sessions [10].

Coffee plays a vital role in Ethiopia's national economy, serving as the country's primary export commodity. Ethiopia is renowned not only as the birthplace of Arabica coffee but also for its premium-quality coffee, celebrated for its unique aroma and flavor [11]. The distinctive characteristics of Ethiopian coffee stem from a combination of geographical factors—such as altitude, soil, temperature, rainfall, and topography—along with diverse genotypes and cultural

practices within the country. Due to this factor, Ethiopian coffee is mainly categorized into nine distinct categories: Yirgacheffe, Sidamo, Harrar, Bebeka, Teppi, Limmu, Jimma, Illubabor, Lekempti, Wellega, Gimbi, Gujj, Yeki, Godere, Anderacha, and Kaffa, and they have their characteristics, brand, and label [12] [13].

2.3.2 Ethiopian Coffee Bean Classification

Ethiopia's traditional coffee-producing regions, such as Yirgacheffe and Sidama, are undergoing a redefinition, with newer, more localized names emerging to capture the specific origins and distinctive characteristics of various coffee varieties [14].

As the birthplace of coffee cultivation, Ethiopia is celebrated for its remarkable diversity of flavors. Ethiopian coffee is classified based on region, altitude, and quality score rather than specific varieties, with estimates indicating the existence of 6,000 to 10,000 unique types [15].

Coffee growing in different parts of Ethiopian regions has genetic variability. Nowadays, the Quality and varieties of grain seeds have been determined manually by visual inspection by Experienced technicians[11].

2.3.3 Ethiopian Coffee Bean Processing

Ethiopian coffee is famous for its distinct flavor and fragrance, with the nation being recognized as the origin of coffee. Ethiopian coffee beans are processed using either the washed or unwashed method [16] [17]. The washed processing method is used in some regions where plenty of fresh water is available. In this method, the cherries are pulped followed by fermentation and washing to remove the cover. The result beans are dried in the sun. In natural processing, unpulsed cherries are dried in the sun. It required more time to dry than washed processing. In Ethiopia, the unwashed coffee product accounts for 80% of the total market[12]

2.4 Digital Image Processing and its Application

Images are ways of recording and presenting information in visual form and, in the broadest sense, an image corresponds to any kind of two-dimensional data[18]. The inherent nature of images is not conducive to computer processing, as computers rely on numerical data rather than visual information for operation. Therefore, images need to be converted into numerical data referred to as digital images, to enable computer manipulation[11]

Digital image processing, also known as computer image processing, refers to the process of converting an image signal into a digital signal and processing it with a computer[18]. Image processing involves a series of steps to improve or extract information from images. These steps include enhancing image quality, reducing noise, dividing images into segments, restoring damaged parts, encoding images for efficient storage, compressing images to save space, and extracting relevant features.

Image processing is a technique that applies various operations to an image to either enhance its visual appearance or extract meaningful information. It can be considered a type of signal processing specifically tailored for images, where the input is an image and the output can be an improved image or a set of characteristics derived from that image. Image processing is a rapidly evolving field with significant applications in engineering and computer science. The typical image-processing pipeline consists of three main stages:

1. Image Acquisition: Importing the image using specialized tools.
2. Image Processing: Analyzing and manipulating the image to achieve the desired outcome.
3. Image Output: Generating the final result, which can be a modified image or a report summarizing the image analysis.

Image processing encompasses two primary methods: analog and digital. Analog image processing is commonly applied to physical formats such as printouts and photographs, where visual techniques guided by interpretation principles are utilized by image analysts [23], [24], [25]. In contrast, digital image processing involves computer-based techniques to modify digital images. This approach typically involves three main stages: pre-processing, enhancement and display, and information extraction [24], [25].

Digital image processing has seen remarkable advancements, significantly influencing numerous fields globally (Reference [19]). Its applications are diverse, spanning areas such as forest surveys, disaster monitoring, resource exploration, and urban planning. For example, in aviation, digital image processing enhances capabilities at organizations like the Jet Propulsion Laboratory (JPL), aiding in the analysis of images obtained from lunar and Martian missions. Additionally, it plays a critical role in aircraft and satellite remote sensing, using reconnaissance aircraft to examine specific regions of the Earth [19].

2.4.1 Image acquisition

Image acquisition refers to the process of capturing visual data from the real world and converting it into a digital format that can be processed, analyzed, stored, and manipulated by computers. This process typically involves the use of cameras or other optical devices to capture images of objects, scenes, or events. There are various methods and devices used for image acquisition, depending on the specific application and requirements. Common devices used for image acquisition include digital cameras, Smartphones, webcams, scanners, and medical imaging equipment like MRI machines or X-ray scanners.

Image acquisition is a crucial step in various fields such as medical imaging, surveillance, remote sensing, astronomy, and digital photography. The quality and fidelity of the acquired images depend on factors like the resolution of the sensor, lens quality, lighting conditions, and the settings of the acquisition device.

2.4.2 Image preprocessing

Image acquisition involves capturing visual data from the physical environment and transforming it into a digital format suitable for computer processing, analysis, storage, and manipulation. This process generally involves optical devices, such as cameras, to capture images of objects, environments, or events. Depending on the application and requirements, various methods and devices are employed for image acquisition. Commonly used devices include digital cameras, smartphones, webcams, scanners, and specialized medical imaging equipment like MRI or X-ray machines. This step is essential across multiple fields, including medical imaging, surveillance, remote sensing, astronomy, and digital photography. The quality and accuracy of the captured images are influenced by factors such as sensor resolution, lens performance, lighting conditions, and device settings.

2.4.3 Image Segmentation

Image segmentation is a key process in computer vision, where an image is divided into multiple segments or regions to simplify its representation and enable easier analysis [26]. The primary objective is to cluster pixels that either belong to the same object or share similar attributes, effectively separating them from the background or other objects. This process is critical for a wide range of applications, including object detection, image recognition, medical image analysis, and scene understanding. Segmentation can be performed at varying levels of detail, from broad

segmentation into larger regions to highly detailed pixel-level segmentation. The selection of a segmentation method is influenced by factors such as image complexity, object characteristics, computational constraints, and the requirements of the specific application [26].

2.4.4 Feature Extraction

Feature extraction is a crucial process in machine learning and computer vision that involves identifying and isolating meaningful information from raw data, such as images, text, or signals. For image classification, this process transforms raw image data into a concise representation that encapsulates significant patterns or characteristics. These representations, often structured as feature vectors, serve as input for machine learning tasks like classification, clustering, or information retrieval. The goal of feature extraction is to emphasize discriminative attributes while reducing the data's dimensionality, thereby enabling more efficient and accurate learning by subsequent models. In deep learning, convolutional neural networks (CNNs) are widely used for automated feature extraction from images, leveraging hierarchical layers to learn increasingly complex and abstract features from raw pixel data.

2.4.5 Classification

Image classification involves identifying and assigning categories to objects based on measurable features. This process utilizes classifiers or models to predict the labels or classes associated with the objects. Two primary approaches to image classification exist supervised and unsupervised methods.

In machine learning, classification is a supervised learning task where the goal is to assign a label or category to input data based on its features. This process requires training a model on a labeled dataset, where each data instance is associated with a specific category or class. Once trained, the model can classify new, unseen data accurately. Classification is widely used in various fields, including finance, healthcare, marketing, and computer vision.

The distinct categories or labels that a classification model predicts are known as classes. In binary classification, the task involves two classes, such as distinguishing between "spam" and "not spam" in an email filtering system. Multi-class classification extends this idea to involve more than two classes, such as categorizing different types of fruits like coffee beans, apples, oranges, and

bananas based on their features. Additionally, multi-label classification allows each instance to be associated with multiple classes simultaneously. This is particularly useful in scenarios where an input might belong to several categories, such as tagging a photo with multiple objects.

2.4.5.1 Naïve Bayes Classifier

Naive Bayes is a group of straightforward yet highly effective probabilistic classifiers grounded in Bayes' Theorem. It operates under the assumption that the features used to predict the target class are conditionally independent, given the class label. This naive independence assumption simplifies calculations and enables efficient training, even when dealing with large datasets [27].

Despite its simplicity, Naive Bayes classifiers perform remarkably well in various applications, particularly in text-based tasks such as spam detection, sentiment analysis, and document classification. The Bayesian classification process can be outlined as follows:

$$P(C/X) = \frac{P(X/C)P(C)}{P(X)} \qquad \text{EQ 2.1}$$

Where:

- P(C|X) is the posterior probability of class C given the evidence X.
- P(X|C) is the likelihood of the evidence X given that the class is C.
- P(C) is the prior probability of class C.
- P(X) is the marginal likelihood or the probability of the evidence X.

2.4.5.2 Artificial Neural Networks (ANNs)

Artificial Neural Networks (ANNs) are a cornerstone of machine learning, modeled after the structure and functioning of the human brain [28]. They are composed of interconnected layers of neurons that process input data and learn to perform tasks such as classification and pattern recognition. Each neuron calculates a weighted sum of its inputs, incorporates a bias term, and applies an activation function to produce an output.

ANNs analyze sensory data using machine perception techniques to label or cluster raw inputs. The patterns recognized by ANNs are numerical, represented as vectors, and serve to translate diverse real-world data—such as text, images, sound, or time series—into mathematical expressions [29]. This process is often expressed as:

$$y = f(\Sigma_{i=1}^{n} wixi + b) \qquad \text{EQ 2.2}$$

Where:

- xi represents the inputs to the neuron.
- wi are the weights associated with each input.
- b is the bias term.
- f(·) is the activation function.
- y is the output of the neuron.

The network typically includes an input layer, multiple hidden layers, and an output layer. During training, the network adjusts its weights and biases using backpropagation and optimization algorithms like Stochastic Gradient Descent (SGD) to minimize a loss function, which measures the difference between the predicted and actual outcomes. ANNs are widely used in applications such as image processing, natural language processing, and healthcare, though they face challenges like over-fitting, high computational demands, and the need for large labeled datasets [30].

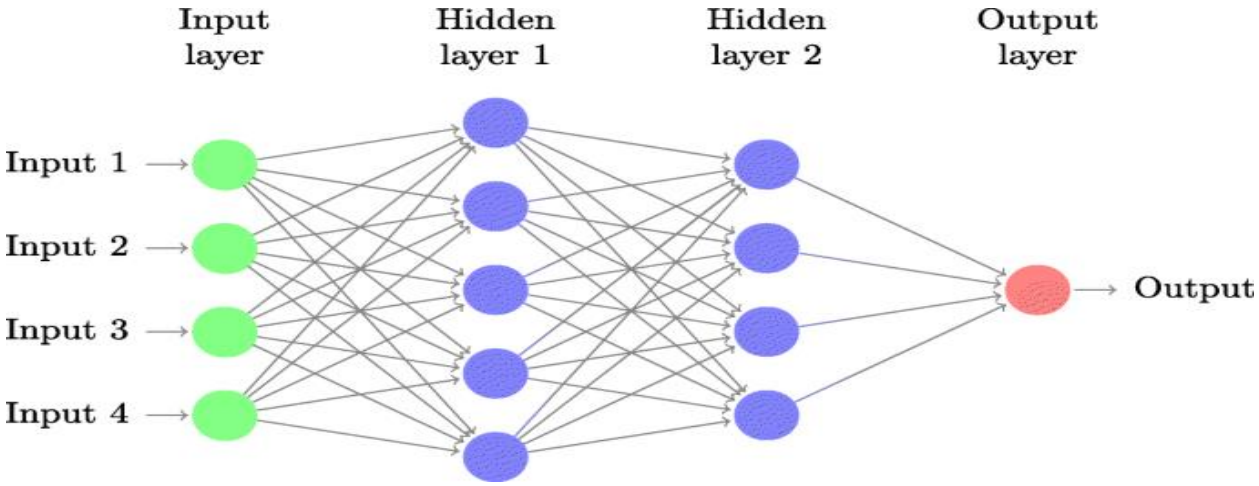


Figure2.1 Structure of Artificial Neural Networks (ANNs)

2.4.5.3 Recurrent Neural Networks (RNNs)

Recurrent Neural Networks (RNNs) are a type of artificial neural network designed for processing sequential data, where the order and context of the data points are important [31]. Unlike feed-forward neural networks, RNNs have connections that loop back on themselves, allowing them to

maintain a form of memory [29]. This architecture enables RNNs to capture temporal dependencies and patterns in sequences, making them particularly useful for tasks such as time series forecasting, speech recognition, and natural language processing.

Mathematically, at each time step t, an RNN updates its hidden state h_t based on the input x_t and the previous hidden state h_{t-1}. This update can be expressed as:

$$h_t = f(W_h h_{t-1} + W_x x_t + b) \qquad \text{EQ 2.3}$$

Where:

- W_h is the weight matrix for the hidden state,
- W_x is the weight matrix for the input,
- b is the bias term,
- $f(\cdot)$ is the activation function, often a non-linear function like tanh or ReLU.

RNNs can suffer from issues such as vanishing and exploding gradients, which can make training difficult, especially for long sequences. Variants like Long Short-Term Memory (LSTM) networks and Gated Recurrent Units (GRU) address these problems by incorporating gating mechanisms that control the flow of information and preserve long-term dependencies more effectively [29].

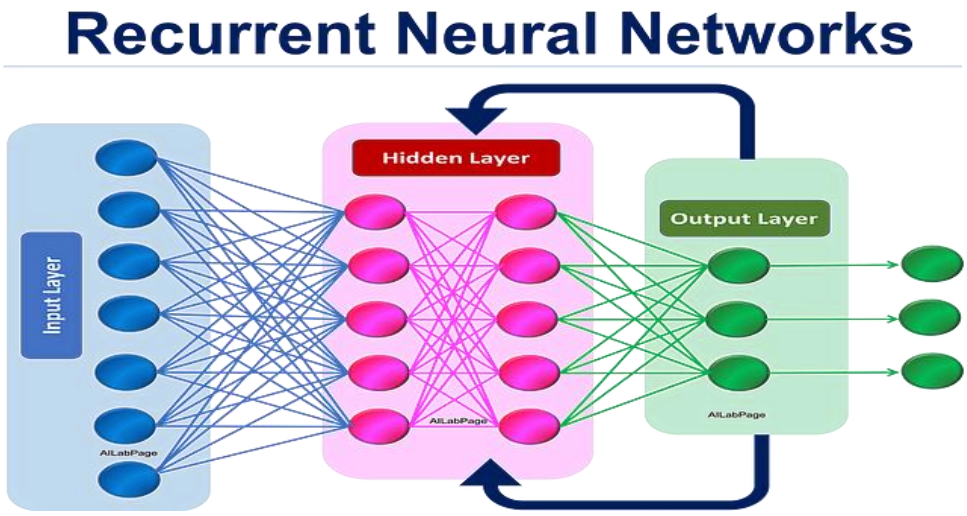


Figure2.2 Structure of Recurrent Neural Networks (RNNs)

2.4.5.4 Deep-learning

Deep learning is a subset of machine learning that focuses on algorithms inspired by the structure and function of the brain, known as artificial neural networks [32]. These algorithms aim to model high-level abstractions in data by utilizing multiple processing layers, or neural networks, which consist of many interconnected nodes, or artificial neurons. Similar to synapses in the human brain, each connection in a neural network transmits signals from one artificial neuron to another [33].

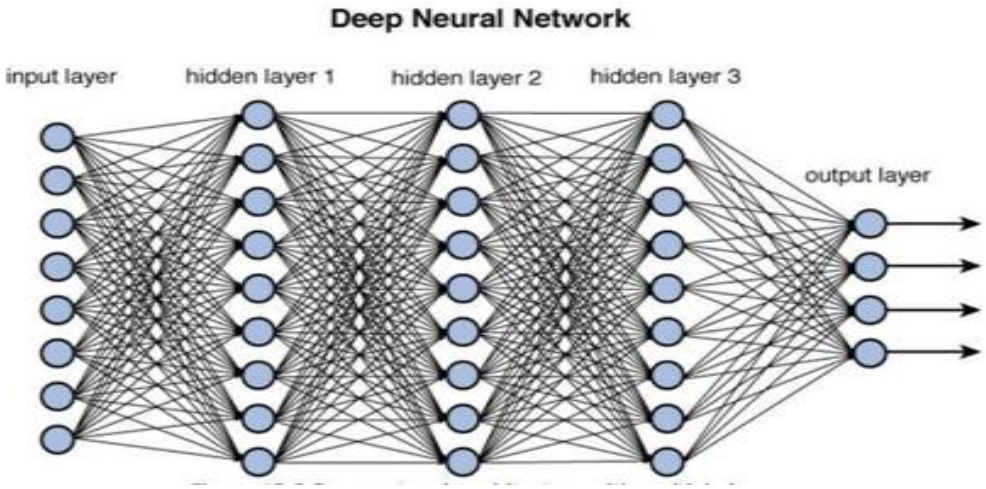


Figure2.3 Deep network architecture with multilayer

Deep learning is a branch of machine learning where computers learn from past experiences, enabling them to understand the world by recognizing concepts organized in hierarchical layers. As the computer acquires knowledge from these experiences, there is no need for a human operator to manually define all the required knowledge [34], [35]. The rise of deep learning can be attributed to its remarkable success in tasks such as image and speech recognition, natural language processing, autonomous vehicles, and more. A key factor in its success is the ability of deep learning models to automatically learn data representations directly from raw inputs, often eliminating the need for explicit feature engineering.

Deep learning enables computational models with multiple processing layers to learn and represent data with multiple levels of abstraction, similar to how the brain processes and understands multimodal information, thus implicitly capturing the intricate structures within large-scale data [36]. It is a broad family of methods that includes neural networks, hierarchical probabilistic models, and various unsupervised and supervised feature learning algorithms [37]. The recent surge in deep learning's popularity is largely due to its demonstrated ability to outperform previous

state-of-the-art techniques across a range of tasks, alongside the growing availability of complex data from diverse sources [36].

Some popular deep learning architectures include Convolution Neural Networks (CNNs) primarily used for image-related tasks, Recurrent Neural Networks (RNNs) for sequential data like text and speech, and Transformers, which have gained prominence in natural language processing tasks. These architectures, along with advancements in optimization techniques, hardware acceleration, and large-scale datasets, have propelled deep learning to the forefront of artificial intelligence research and application.

2.4.5.4.1 Convolution Neural Networks

Convolution Neural Networks (CNNs) were inspired by the visual system's structure, and in particular by the models it proposed. The first computational models based on this local connectivity between neurons and on hierarchically organized transformations of the image are found in Neocognitron, which describes that when neurons with the same parameters are applied on patches of the previous layer at different locations, a form of translational invariance is acquired[38]. Yann LeCun and his collaborators later designed Convolution Neural Networks employing the error gradient and attaining very good results in a variety of pattern recognition tasks [36].

Convolutional Neural Networks (CNNs) are deep learning algorithms designed to process input images by applying convolution operations with filters or kernels to extract features. For a given NxN image, this convolution process involves applying a fXf filter, which effectively learns the same feature across the entire image [39]). CNNs are composed of three main types of layers convolutional, pooling, and fully connected layers [38]. The convolution and pooling layers focus on feature extraction, while the fully connected layer translates these features into the final output, such as a classification. The convolution layer in CNNs performs key operations, such as convolution and other specialized linear transformations [38]. In digital images, pixel values are stored in a two-dimensional (2D) grid [33].

A CNN classifier consists of two parts [12].

1. **Feature extraction:** - this part of the CNN extracts features from the given input automatically. The extracted features are called feature maps. Feature extraction consists of convolutional layers

followed by polling layers and activation functions such as ReLU, Tanh, sigmoid, etc. The extracted features are the inputs to the second part.

2. **Classification: -** here the classification process is done. The classification part includes a few fully connected layers complemented by a loss function.

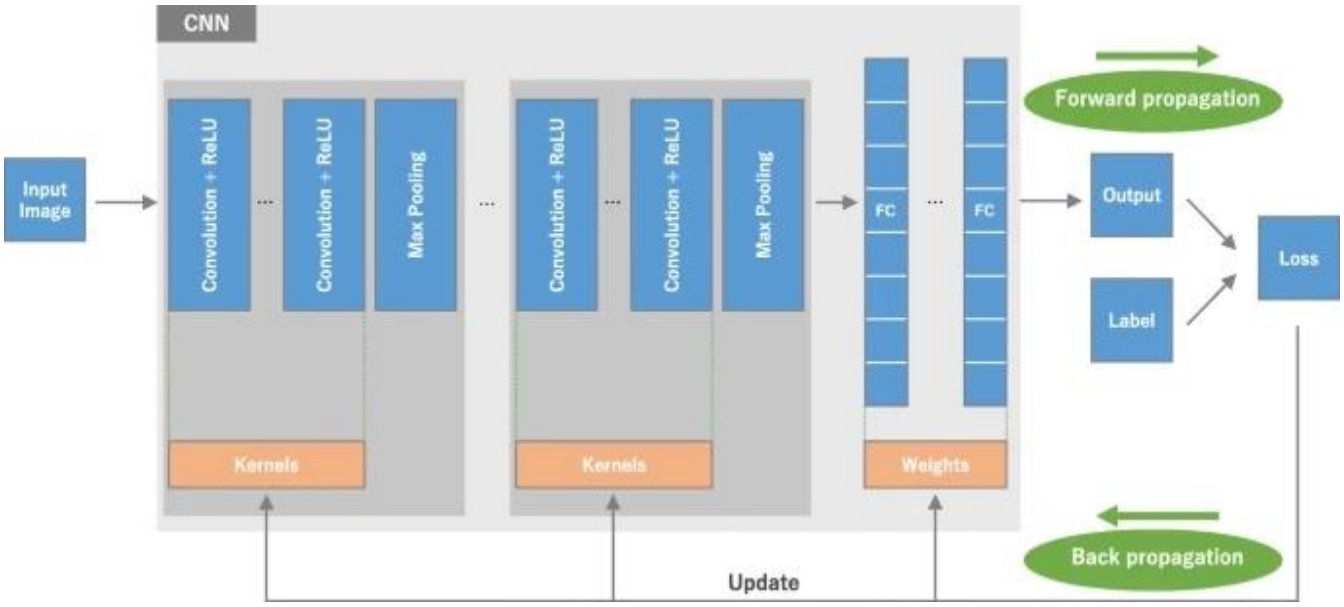


Figure 2 4 Architecture of Convolution Neural Networks [48]

2.4.5.4.1.1 Convolution Layer

A convolutional layer is a fundamental building block in Convolutional Neural Networks (CNNs), a class of deep learning models commonly used for image processing tasks. The convolutional layer performs a convolution operation on the input data, which is typically an image or a feature map from a previous layer.

Here's how a convolutional layer works:

Convolution operation: The convolutional layer applies a set of filters (also known as kernels) to the input data. Each filter is a small matrix of weights that slides over the input data spatially [30]. At each position, the filter computes the dot product between its weights and the corresponding region of the input data. This computation produces a single value, which forms a new pixel in the output feature map.

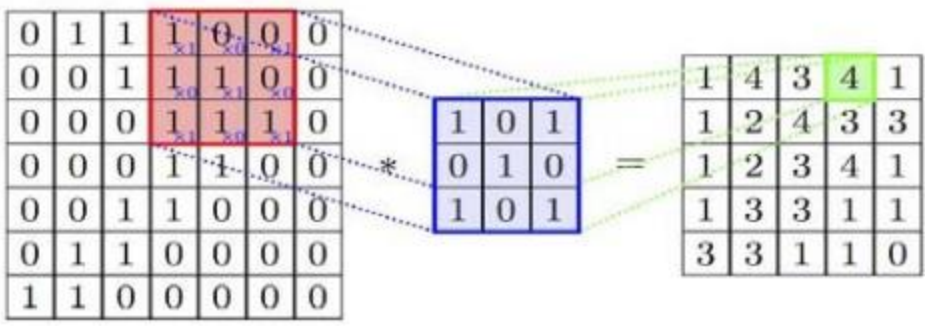


Figure 2.5 Convolution operation [50]

The formula for the convolution operation in a Convolutional Neural Network (CNN) involves applying a filter (also known as a kernel) to the input image to produce a feature map. Here's the formula for a 2D convolution operation:

Given:

- **Input image**: *I* (with dimensions *Win*×*Hin*×*Din*, where *Win* is the width, *Hin* is the height, and *Din* is the number of input channels)
- **Filter/kernel:** *K* (with dimensions *F*×*F*×*Din*, where *F* is the filter size)
- **Stride**: *S* (the number of pixels to shift the filter/kernel across the input image)
- **Padding:** *P* (the number of pixels added to the input image border)

The formula for the 2D convolution operation at a specific spatial location (*x*,*y*) in the output feature map is:

$$\mathrm{Oi,j,k} = \sum_{l=0}^{Din-1}.\ \sum_{m=0}^{F-1}.\ \sum_{n=0}^{F-1} \mathrm{I\ (i\ x\ S + l),(j\ x\ S + m),l\ x\ Km,n,l,k} \qquad \text{Eq (2.4)}$$

Where:

- ***Oi, j, k*** is the value at spatial location **(*i,j*)** in the output feature map at channel *k*.
- ***I*(*i*×*S*+*l*),(*j*×*S*+*m*),*l*** is the pixel value in the input image at spatial location **((*i*×*S*+*l*,*j*×*S*+*m*)** and channel ***l*.**
- ***Km,n,l,k*** is the value of the filter/kernel at position ***(m,n)*** for input channel *l* and output channel ***k.***

After computing ***Oi, j,k*** for all spatial locations ***(i,j)*** and output channels ***k,*** we obtain the complete output feature map ***O.***

This formula represents a single convolutional operation. During the convolution operation, the filter/kernel is slid across the input image with the specified stride, and the element-wise multiplication and summation are performed as shown in the formula to compute the output feature map. Padding can also be applied to control the spatial dimensions of the output feature map.

2.4.5.4.1.2 Pooling Layers

The Pooling layer involves extracting a specific value from a set of values, typically the maximum or average value among them. This process effectively reduces the size of the output matrix. For instance, in max-pooling, the maximum value is selected from a 2 x 2 segment of the matrix. This operation captures the information indicating the presence of a feature in that particular part of the image. By doing so, unnecessary details regarding feature presence in a specific image section are discarded, focusing only on essential information. It is a common practice to intermittently insert a Pooling layer between consecutive convolutional blocks in CNN architecture. The primary purpose is to gradually diminish the spatial size of the representation, thereby decreasing the number of parameters and computational load in the network.

Max Pooling:

A type of pooling operation commonly used in Convolutional Neural Networks (CNNs) to down-sample feature maps while retaining the most significant information. In max pooling, the maximum value within each local region of the input feature map is retained, while other values are discarded. This helps in capturing the most prominent features while reducing the spatial dimensions of the feature maps. The formula for max pooling:

Given:

- Input feature map: $\boldsymbol{I}$ (with dimensions $\boldsymbol{Win \times Hin \times Din,}$ where $\boldsymbol{Win}$ is the width, $\boldsymbol{Hin}$ *i*s the height, and $\boldsymbol{Din}$ is the number of input channels)
- Pooling window size: $\boldsymbol{F \times F}$ (typically square, where $\boldsymbol{F}$ is the size of the window)
- Stride: S (the number of pixels to move the pooling window)

The formula for max pooling at a specific spatial location $\boldsymbol{(x,y)}$ in the output feature map is:

$$O_{i,j,k} = \max_{m=0}^{F-1} \max_{n=0}^{F-1} I_{(i\times S+m),(j\times S+n),k}$$ **Eq (2.5)**

- ***Oi, j, k* is** the value at spatial location ***(i,j)*** in the output feature map at channel ***k*.**
- ***I(i×S+m),(j×S+n),k*** is the pixel value in the input feature map at spatial location ***(i×S+m,j×S+n)*** and channel ***k.***

After applying the max pooling operation with the specified window size and stride, we obtain the down-sampled output feature map O. The max pooling operation retains the maximum value within each pooling window, effectively reducing the spatial dimensions of the feature maps while preserving the most important features.

Average Pooling:

Average pooling is a type of pooling operation commonly used in Convolutional Neural Networks (CNNs) to down-sample feature maps while retaining important information. Unlike max pooling, which retains the maximum value within each local region, average pooling computes the average value of all elements within each local region. Instead of taking the maximum value, average pooling computes the average of all values in each region. This can help in preserving a smoother representation of the input.

The formula for average pooling:

Given:

- Input feature map: **I** (with dimensions **Win×Hin×Din,** where **Win** is the width, **Hin** is the height, and **Din** is the number of input channels)
- Pooling window size: **F×F** (typically square, where ***F*** is the size of the window)
- Stride: **S** (the number of pixels to move the pooling window)

The formula for average pooling at a specific spatial location **(*x*, *y*)** in the output feature map is:

$$\mathrm{Oi,\ j,\ k} = \frac{1}{F^2} \sum_{m=0}^{F-1} . \sum_{n=0}^{F-1} \mathrm{I}\ (\mathrm{i} + \mathrm{S} + \mathrm{m}), (\mathrm{j} + \mathrm{S} + \mathrm{n}), \mathrm{k}$$ Eq (2.6)

Where:

- **Oi, j, k** is the value at spatial location **(i,j)** in the output feature map at channel *k*.

- **I(i×S+m),(j×S+n),k** is the pixel value in the input feature map at spatial location **(i×S+m,j×S+n)** and channel **k**.

After applying the average pooling operation with the specified window size and stride, we obtain the down-sampled output feature map **O**. The average pooling operation computes the average value within each pooling window, effectively reducing the spatial dimensions of the feature maps while preserving the general trends in the data.

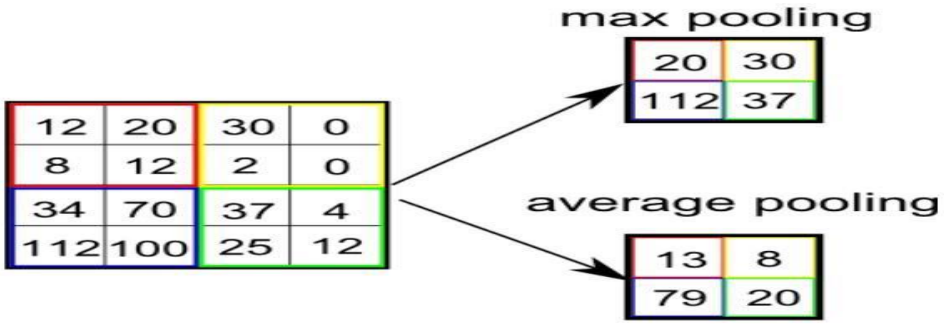


Figure 2. 6 Max and Average Pooling [51]

2.4.5.4.1.3 Activation Function

Activation functions are crucial components of neural networks, including Convolutional Neural Networks (CNNs). They introduce non-linearity into the network, allowing it to learn complex patterns and relationships in the data.

The primary function of an activation function in any neural network model is to map the input to the output. The input value is calculated by taking the weighted sum of the neuron's inputs and adding a bias term (if present). In simple terms, the activation function determines whether a neuron should "fire" or not by producing the corresponding output based on the input it receives. In CNN architectures, non-linear activation layers are applied after each learnable layer, such as convolutional and fully connected (FC) layers. These non-linear layers allow the CNN model to learn more complex patterns and map inputs to outputs in a non-linear fashion. A key feature of an activation function is its differentiability, which is necessary for error backpropagation to effectively train the model [41]. The most commonly used activation functions in deep neural networks, including CNNs, are listed and described below.

Sigmoid

The sigmoid activation function, typically represented as σ(x), is a non-linear function frequently used in neural networks, especially in the output layer for binary classification problems.

The sigmoid function has the following mathematical expression:

$$\alpha(x) = \frac{1}{1+ e^{-x}} \qquad \text{Eq (2.7)}$$

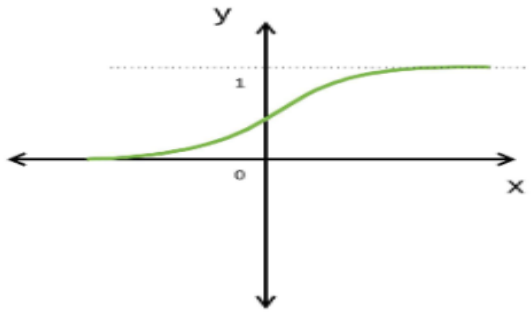


Figure 2. 7 Sigmoid Activation function

Where:

- **x** is the input to the function.
- **e** is the base of the natural logarithm (approximately equal to 2.71828).

The sigmoid function squashes the input values to the range (0, 1), which can be interpreted as probabilities. It maps any real-valued number to the range of 0 and 1. Specifically, when the input is negative, the output is close to 0, and when the input is positive, the output is close to 1.

Hyperbolic Tangent (tanh)

The hyperbolic tangent function, often denoted as tanh tanh(x), is another non-linear activation function commonly used in neural networks. It's similar to the sigmoid function but with outputs ranging from -1 to 1. The mathematical expression for the hyperbolic tangent function is:

$$\tanh(x) = \frac{e^{x}- e^{-x}}{e^{x}+ e^{-x}} \qquad \text{Eq(2.8)}$$

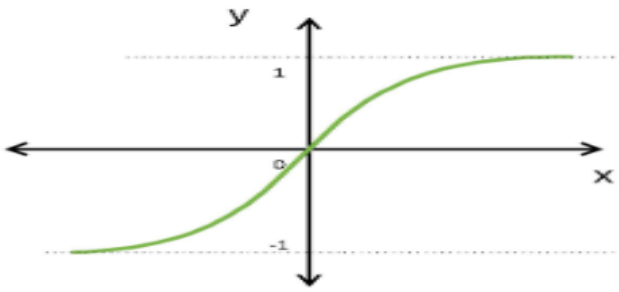


Figure 2. 8 Hyperbolic Tangent (tanh) function

Where:

- **x** is the input to the function.
- **e** is Euler's number, approximately equal to 2.71828.

The hyperbolic tangent function is commonly used in neural networks for various tasks, including classification, regression, and recurrent neural networks (RNNs). It shares some similarities with the sigmoid function but has a broader range of outputs, making it particularly useful in scenarios where the input data can have both positive and negative values.

Rectified Linear Unit (ReLU)

The Rectified Linear Unit (ReLU) is the most widely used activation function in Convolutional Neural Networks. It works by converting all input values to positive numbers. One of the main advantages of ReLU is its minimal computational load compared to other activation functions.[41].

Mathematically, the **ReLU** function is defined as:

$$ReLU(x) = \max(0, x) \quad \text{Eq (2.9)}$$

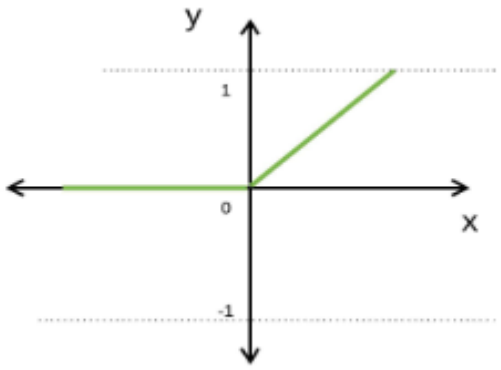


Figure 2.9 Rectified Linear Unit (ReLU)

Leaky ReLU

In contrast to ReLU, the Leaky ReLU activation function does not completely ignore negative inputs; instead, it downscales them. Leaky ReLU is designed to address the "Dying ReLU" problem, where neurons can become inactive and stop learning due to ReLU's zeroing out of negative values. Mathematically, the Leaky ReLU function is defined as:

$$Leaky\ ReLU(x) = \begin{cases} x, & if\ x > 0 \\ ax, & otherwise \end{cases} \qquad \text{Eq (2.10)}$$

Where:

- ***x*** is the input to the function.
- ***α*** is a small positive constant (typically around 0.01) that determines the slope of the function for negative inputs.

The Leaky ReLU function preserves the positive output of the ReLU function for positive inputs **(*x*>0),** while for negative inputs ($x \leq 0$), it multiplies the input by the small positive constant α instead of outputting zero. This small slope allows gradients to flow even when the input is negative, addressing the issue of neurons dying out during training.

Softmax

Softmax is a widely used activation function, particularly in the output layer of neural networks used for multi-class classification tasks. It's used to transform the raw output scores of a neural network into probabilities. The Softmax function takes as input a vector of real-valued scores

(often called logics) and normalizes them into a probability distribution over multiple classes. The formula for the softmax function for a vector *z* of *K* real numbers is given by:

$$softmax(zi) = \frac{e^{2j}}{\sum_{j=1}^{z} e^{2j}} \qquad \text{Eq (2.11)}$$

Where:

- ***zi*** is the ***i***-th element of the input vector ***z***.
- ***e*** is Euler's number, approximately equal to 2.71828.
- The sum in the denominator is over all elements of the input vector ***z***.

The Softmax function exponentiates each element of the input vector, which makes all values positive, and then divides each exponentiates value by the sum of all exponentiates values. This normalization ensures that the resulting values lie between 0 and 1 and sum up to 1, making them interpretable as probabilities.

2.4.5.4.1.4 Fully Connected Layer

The final convolution layer's output is flattened, converting it into a one-dimensional array, and then linked to one or more fully connected layers. Following the convolution layers and pooling stages for feature extraction, the features are processed by a series of fully connected layers to produce the ultimate outputs of the network before passing through the classifier. The number of nodes in the last fully connected layer corresponds to the number of classes being classified. In a fully connected layer, each neuron or node is connected to every neuron in the preceding layer, hence the term fully connected. These connections are weighted, and each connection has its weight parameter. The output of each neuron in the fully connected layer is computed by taking a weighted sum of the inputs from the previous layer, adding a bias term, and then passing the result through an activation function.

Mathematically, the output ***yi*** of the ***i*****th** neuron in a fully connected layer can be expressed as:

$$yi = f\left(\Sigma_{j=1}^{n} w\text{ij} \,.\, \text{xj} + \text{bi}\right) \qquad \text{Eq (2.12)}$$

Where:

- ***Yi*** **is** the output of the ***i*****th** neuron.
- ***f*** is the activation function applied element-wise.

- ***wij*** is the weight of the connection between the ***j*th** neuron in the previous layer and the ***i*th** neuron in the current layer.
- ***X j*** is the output of the ***j*th** neuron in the previous layer.
- ***Bi*** is the bias term associated with the ***i*th** neuron.
- ***n*** is the number of neurons in the previous layer.

Fully connected layers are often used in the final stages of neural networks for tasks such as classification, where the output of the network is produced by these fully connected layers after feature extraction and transformation through earlier layers. However, in deep learning architectures, fully connected layers may be preceded by other layer types, such as convolutional layers or recurrent layers, which extract hierarchical features from the input data.

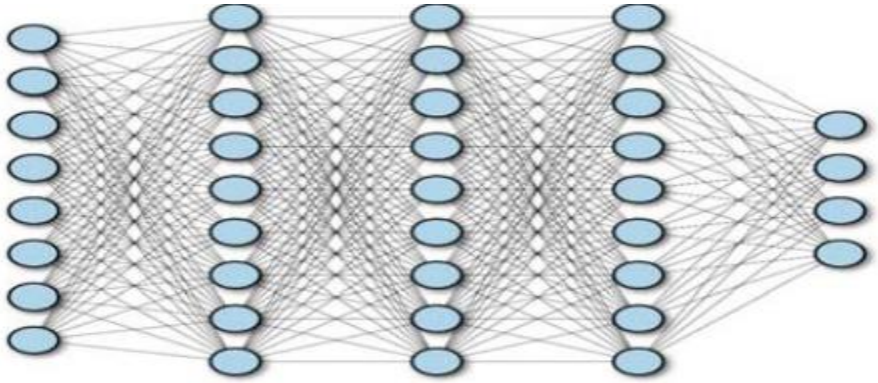


Figure 2.10 Fully connected layers

Dropout

Dropout is a regularization technique commonly used in neural networks, including fully connected layers, to prevent over-fitting. Over-fitting occurs when a model learns to memorize the training data too well, including noise and random fluctuations, resulting in poor generalization to unseen data. Dropout helps mitigate over-fitting by introducing noise to the network during training. Here's how dropout works:

Dropout is a simple yet powerful regularization technique that has been shown to improve the generalization performance of neural networks, especially in situations where the training data is limited or prone to over-fitting. It is widely used in practice and has become a standard component in many neural network architectures.

During Training: For each training example and each iteration (or mini-batch) of training, dropout randomly "drops out" (sets to zero) a fraction of neurons in the fully connected layer with a predefined probability, typically denoted as **p**. This means that the output of these neurons is ignored, and their connections are temporarily removed from the network for that iteration.

During Testing: During testing dropout is typically turned off, and all neurons are used. However, the weights of the neurons are scaled by **1−p** to account for the fact that more neurons are active during testing than during training.

By randomly dropping out neurons during training, dropout prevents neurons from relying too heavily on the presence of other specific neurons. This encourages the network to learn more robust features that are not overly dependent on the presence of any particular neuron. Dropout effectively creates an ensemble of multiple neural network architectures, as different subsets of neurons are dropped out during each training iteration.

The dropout rate **p** is a hyper-parameter that needs to be tuned during model training. Common values for dropout rates range from 0.2 to 0.5, but the optimal value may vary depending on the dataset and the architecture of the neural network.

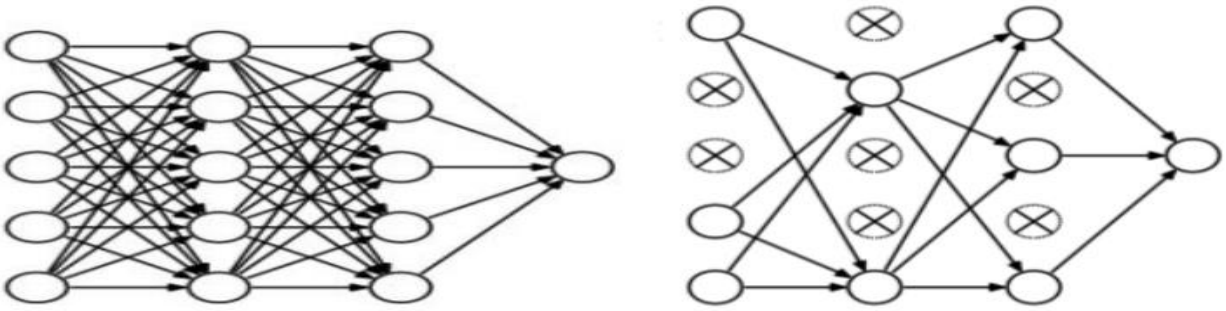


Figure 2.11 Dropout method to prevent over-fitting [53]

a) Standard network b) After applying dropout

2.5 Related Works

Numerous research studies have utilized image processing techniques in conjunction with various learning-based classifier algorithms to categorize coffee beans, primarily for marketing purposes. Deep learning methodologies have also been employed in the classification and identification of

various edible products. The literature review reveals several studies focused on the classification of different varieties of edible products. Our review primarily concentrates on previous research conducted on the classification of varieties in other crops. Below are examples of prior literature investigated by various researchers. However, it's important to note that our research focuses on the classification of coffee varieties.

H. M. Aycheh [17] The researcher aims to develop a digital image analysis technique for classifying different varieties of Ethiopian coffee based on their growing regions. Coffee samples were collected from six popular coffee-growing regions in Ethiopia: Bale, Harar, Jimma, Limu, Sidamo, and Welega. A total of 309 images containing 4844 coffee beans were captured, with an average of 56 images per region. The processing type of the coffee (washed or unwashed) was also considered during the analysis. Ten morphological features and six color features were extracted from each coffee bean image. These features were used to characterize the shape and color of the coffee beans, respectively.

The study compared the classification performance of two classifiers: Naïve Bayes and Neural Networks. The classifiers were evaluated based on their ability to classify coffee beans according to their morphology, color, and a combination of both features. The study found that the Neural Network classifier outperformed the Naïve Bayes classifier in terms of classification accuracy. Additionally, it was observed that morphology features had better discrimination power compared to color features. However, when both morphology and color features were used together, the classification accuracy improved significantly. The best classification accuracies were achieved using Neural Networks with both morphology and color features combined Bale 80.7%, Harar 72.6%, Jimma 56.8%, Limu 96.77%, Sidamo 95.42%, Welega 69.9%.The overall classification accuracy across all regions was 77.4%.

G. Amtate and D. Teferi [12] in this research described an effective application of deep learning techniques, specifically Convolutional Neural Networks (CNN), for the classification of Ethiopian coffee beans based on their geographical origins. The primary goal of the research is to develop a model that can accurately characterize and identify coffee beans from six different regions of Ethiopia: Jimma, Limmu, Nekemte, Yirgacheffe, Bebeka, and Sidama. This involves using image processing techniques and deep learning algorithms. The researchers collected a dataset of 9836 coffee bean images from the Ethiopian Coffee Quality and Inspection Center (ECQIC). These images were then used to train, validate, and test a CNN model. The dataset was split into 70% for

training, 10 for validation, 10 for testing 80% for training, 10% for validation, and 10% for testing. Various hyperparameters and dataset sizes were experimented with to optimize the model's performance. The CNN model achieved an impressive overall classification accuracy of 99.89% and a generalization log loss of 0.92%. This indicates that the model is highly effective in accurately classifying Ethiopian coffee beans based on their geographical origins.

A. B. Emiru [27] The researcher described focuses on the automatic identification of faba bean varieties using different local descriptors and deep learning approaches. The proposed model was implemented using Keras with TensorFlow backend in Python, utilizing a sample dataset collected from Adet Agricultural Center. The model consists of four main components: preprocessing, segmentation, feature extraction, and identification. Two approaches were followed for faba bean variety identification:

First Approach Different local descriptor features such as HOG, SURF, LBP, and GLCM were combined, and PCA dimensionality reduction methods were applied to train Random Forest (RF), Support Vector Machine (SVM), and Multilayer Perception (MLP) classifiers.

Second Approach Convolutional Neural Networks (CNN) were utilized, where adaptively learned features and their levels were employed to identify images. Various local descriptors were aggregated with CNN models to achieve state-of-the-art results, particularly when the dataset size was small. The CNN model achieved 98.67% training accuracy and 92.46% testing accuracy.

D. Buonocore, M. Carratù, and M. Lamberti [42] This paper introduces a Convolutional Neural Network (CNN) approach for coffee bean detection and grains error classification, with a focus on automatically classifying coffee bean species via images. The primary goal was to develop a CNN-based approach for accurately detecting coffee bean species and classifying grain errors. In this paper, the proposed method utilizes the YOLO (You Only Look Once) algorithm, a popular object detection technique, for coffee bean detection and classification. The CNN model was trained on a dataset of coffee bean images to learn to accurately distinguish between different coffee species. The study concludes that object detection techniques, specifically the CNN approach utilizing the YOLO algorithm, can effectively classify coffee bean species and contribute to combating food fraud. The results highlight the potential of using such techniques in various applications related to coffee quality control and assurance.

B. Turi, G. Abebe, and G. Goro [52] the researcher was described as focusing on developing a computer routine algorithm to characterize Ethiopian coffee beans from four distinct growing

regions: Hararghe, Jimma, Wollega, and Yirgacheffe. The researcher aimed to create an accurate and objective method for classifying coffee beans based on their geographic origins, as traditional visual inspection methods are subjective and prone to error. Artificial Neural Network (ANN) was employed for classification. Four classification setups were used based on the features used color, morphology, texture, and a combination of morphology and color. A total of 160 images (40 from each location) of coffee bean samples from four regions were used. Important features such as color, morphology, and texture were extracted from these images. In total, 29 features (11 colors, 6 morphological features, and 12 textural features) were extracted. Out of 80 sample images, 70% were used for training (56 images), 25% for testing (20 images), and 5% for validation (4 images). The classification scores achieved were 95% for color, 100% for morphology, 87.5% for texture, and 100% for the combination of morphology and color features. It was found that morphology and the combination of morphological and color features exhibited the highest accuracy.

T. Debela [11] This research work focuses on using a digital image analysis technique to classify Wollega coffee beans (WCBs) genotypes based on their color and morphological features. The study also evaluates the performance of Artificial Neural Network (ANN) in correctly classifying WCBs grown in Haru Coffee Research Center to their origin of the collection. The objective was to classify WCB genotypes based on color and morphological features and evaluate the performance of ANN in correctly identifying the origin of WCBs grown in a specific research center. The study employs three classification set-ups classification based on color feature, morphological feature, and a combination of color and morphological features. The features of 500 images of the WCBs genotype are used as inputs for the ANN. Out of the 500 images, 60% (300 images) were used for training the ANN, 20% (100 images) for testing, and 20% (100 images) for validation. The accuracy of classification using color, morphological, and a combination of color and morphological features are 69%, 71%, and 69% respectively.

C. Pinto, J. Furukawa, H. Fukai, and S. Tamura [43] The study conducted in Timor-Leste aims to develop an automatic coffee bean sorting system to enhance the value of coffee efficiently at the local production level. The initial step involves developing an image processing system to classify images of green coffee beans into different types of defects. Deep Convolutional Neural Networks (CNN), a state-of-the-art machine learning technique, was employed for this purpose. The study utilizes image processing techniques to classify images of green coffee beans into various types of defects. Deep Convolutional Neural Networks (CNN) was employed as the machine learning

technique for image processing and classification. The research successfully achieved sorting accuracy ranging from 72.4% to 98.7% based on the types of defects present in the coffee beans. This indicates the effectiveness of the developed image processing system in accurately identifying and sorting defective coffee beans.

J. N. C. Sarino, M. M. Bayas [44] The study aims to classify the degree of roasting of coffee beans using image processing and artificial neural networks, aiming to improve upon the subjective nature of human inspection. Traditionally, human inspectors assess the degree of roast based on sight and touch, which could be influenced by external factors such as lighting conditions, fatigue, and emotions. By employing image processing techniques, the study seeks to eliminate these external influences and improve the accuracy of roast-level detection. The study focuses on Excelsa coffee beans from Indang, Cavite, and uses a Smartphone to capture images of the roasted beans. Image features, specifically the red (R), green (G), and blue (B) components, were extracted from the images. Artificial neural networks were then employed to classify the coffee beans into three roast levels: light roast, medium roast, and very dark roast. The RGB values serve as input features for the neural network. The proposed method achieves a high accuracy of 97.22% in identifying the degree of roasting of the coffee beans. This indicates that the combination of image processing and artificial neural networks effectively distinguishes between different roast levels based on RGB values extracted from the images.

J. Y. Kim [45] The study introduces a machine learning model, specifically Random Forest, for predicting coffee quality, to support coffee-producing countries in maintaining competitiveness in the market and contributing to the sustainable development of society. This study explored the importance of the coffee industry in countries including Brazil, Vietnam, Colombia, Indonesia, Ethiopia, Honduras, India, Peru, and Uganda. The researchers likely collected data on various parameters related to coffee quality, such as bean size, color, flavor profile, and perhaps environmental factors like altitude and soil quality. They then trained a Random Forest model using this data to predict the quality of coffee beans. The researcher likely collected data on various parameters related to coffee quality, such as bean size, color, flavor profile, and perhaps environmental factors like altitude and soil quality. They trained a random forest model using this data to predict the quality of coffee beans. The study reports an F1 score of 61.7% for the Random Forest model. The F1 score was a metric that combines precision and recall, providing a measure of the model's accuracy in predicting both positive and negative instances.

M. S. Fuentes, N. A. L. Zelaya [46] The research focused on addressing challenges faced by coffee producers in Honduras, particularly concerning labor shortages and the need for efficient harvesting methods. The aim is to develop a system that can detect and classify coffee fruits as ripe or not ripe, thereby reducing costs, and time, and improving the quality of the final product. The research adopts a qualitative approach with an experimental design. An algorithm was developed to classify coffee fruits based on their ripeness. The deep learning algorithm is trained using a dataset of 196 images, including 108 positive (ripe) and 88 negative (not ripe) examples. The system demonstrates high accuracy, correctly classifying 41 out of 42 test cases, resulting in an efficiency of 97.6%. This indicates that the system effectively identifies ripe and not-ripe coffee fruits, enabling producers to accelerate the harvesting process and improve quality.

S. Wallelign, M. Polceanu [47] The research aims to design a robust Convolutional Neural Network (CNN) model capable of classifying raw coffee beans into their 12 quality grades using small datasets with high data variability. Preprocessing techniques are applied to the input data to reduce irrelevant features. However, the addition of preprocessing techniques does not improve the performance of the CNN model on the dataset. The researcher successfully classifies raw coffee beans into their quality grades with an accuracy of 89.01% on the test dataset.

Table 2.1 summarizes some of the related works

Author(s)	Method	Finding	Limitation
H. M. Aycheh [17]	Naïve Bayes and Neural Networks; Morphology & Color Feature Analysis	Neural Network outperformed Naïve Bayes. Best accuracy (96.77%) achieved using both features; overall accuracy: 77.4%.	Limited regions and feature types; lower accuracy for some regions.
G. Amtate, D. Teferi [12]	Convolutional Neural Networks (CNN)	CNN achieved 99.89% accuracy in classifying beans from 6 Ethiopian regions; highly optimized model.	High performance but requires large datasets; only geographical classification explored.
A. B. Emiru [27]	Local Descriptors (HOG, SURF, LBP, GLCM), PCA, Random Forest, CNN	CNN achieved 98.67% training accuracy, 92.46% testing accuracy for faba bean identification.	Focuses on faba beans, not coffee; performance depends on dataset size.
D. Buonocore et al. [42]	YOLO Algorithm; Convolutional Neural Networks	Effectively classified coffee species and grain errors; highlighted potential	Focused on error classification; limited details on accuracy

		applications in quality control.	metrics for individual species.
B. Turi et al. [52]	Artificial Neural Networks (ANN); Morphology, Color, Texture Analysis	Achieved 100% accuracy with morphology and combined features; 95% with color alone.	Small dataset size; limited to four regions.
T. Debela [11]	ANN; Color and Morphological Feature Analysis	Achieved accuracies of 71% (morphology), 69% (color), 69% (combined); evaluated Wollega coffee bean genotypes.	Moderate accuracy; limited to one region and specific features.
C. Pinto et al. [43]	Deep CNN for Defect Detection and Sorting	Sorting accuracy ranged from 72.4% to 98.7% based on defect types.	Focused on defect detection; lacks geographical classification.
J. N. C. Sarino et al. [44]	ANN; RGB Feature Analysis	Achieved 97.22% accuracy in classifying roast levels of coffee beans using RGB values.	Limited to roast level classification; small dataset size.
J. Y. Kim [45]	Random Forest; Quality Prediction	Predicted coffee quality with an F1 score of 61.7%; explored quality parameters like bean size, color, and flavor.	Moderate prediction accuracy; does not use image processing techniques.
M. S. Fuentes et al. [46]	Deep Learning Algorithm for Ripeness Detection	Achieved 97.6% accuracy in detecting ripe vs. unripe coffee fruits; improved harvest efficiency.	Small dataset; focused only on ripeness detection.
S. Wallelign, M. Polceanu [47]	CNN with Preprocessing	Classified raw coffee beans into 12 quality grades with 89.01% test accuracy.	High variability in data; preprocessing did not improve model performance.

This table organizes the research studies by summarizing the main features used, the technology applied, and the outcomes achieved.

2.5.1 Summary

The reviewed studies focus on various methods and models used in the classification of coffee beans, particularly from Ethiopian regions, employing digital image analysis and machine learning techniques. These studies primarily aim to classify coffee beans based on factors such as geographical origin, morphological features, color, texture, and roasting degree.

1. **Machine Learning Models**: A range of machine learning models, including Naïve Bayes, Artificial Neural Networks (ANN), Convolutional Neural Networks (CNN), and Random Forest, have been applied across the studies. CNNs, in particular, have shown high accuracy in coffee bean classification, especially when combined with deep learning approaches like YOLO for object detection and image processing.
2. **Datasets**: The datasets used vary significantly in size and scope, with some studies relying on relatively small datasets from specific regions or research centers, while others utilize larger, more diverse datasets. The studies commonly collected data from key Ethiopian coffee-growing regions, but the sample sizes and diversity of data sources differ, which impacts the generalizability of the models.
3. **Classification Accuracy**: The classification accuracy achieved by these models ranges from moderate to very high, with CNNs generally outperforming other models in most cases. However, the effectiveness of these models often depends on the specific features used (e.g., color, morphology) and the quality of the dataset.
4. **Limitations**: Several limitations are noted across the studies. These include the restricted geographic scope of the datasets, small sample sizes, and the computational intensity of training complex models like CNNs. Additionally, while some models achieved high accuracy, their performance might not generalize well to other regions or more diverse datasets. Another common limitation is the focus on specific features, such as morphology or color, which may not capture the full complexity of coffee bean classification. Furthermore, none of the previous research has addressed multi-label coffee bean classification.

   In summary, while the reviewed studies demonstrate the potential of machine learning and deep learning models in coffee bean classification, they also highlight the challenges associated with dataset diversity, model generalization, and feature selection. These factors must be carefully considered when developing robust models for coffee bean classification.

CHAPTER THREE

System Design and Modeling

3.1 Overview

System Design and Modeling is the process of conceptualizing and structuring a system to meet specified requirements, considering various factors such as functionality, performance, reliability, and scalability. It involves creating a blueprint that outlines the architecture, components, and interactions of the system, as well as modeling the data and algorithms necessary for its operation. This phase typically includes tasks such as requirements analysis, conceptual design, algorithm selection, system architecture design, data modeling, model development, validation, and iterative refinement. The ultimate goal of system design and modeling is to develop a well-designed and efficient system that effectively addresses the needs and objectives of its stakeholders.

3.2 Data Collection

Data collection for deep learning involves gathering a diverse dataset that accurately represents the target domain. In this case, we're collecting images of coffee beans from four distinct regions: Yirgacheffe, Sidama, Gujii, and Arisi. These regions are known for their unique coffee cultivation practices, each contributing distinct characteristics to the beans they produce. we're including beans graded by the Hawassa Product Quality Inspection and Certification Center in the Sidama region of Ethiopia. This center plays a crucial role in assessing and certifying the quality of coffee beans produced in Sidama. By including beans certified by this institution, we ensure that our dataset covers a wide range of quality grades, reflecting the variability in coffee bean quality. Our dataset specifically focuses on coffee beans produced in the 2016 E.C. (Ethiopian calendar) production year, ensuring that our data is up-to-date and relevant. However, collecting such data comes with challenges, such as variations in lighting conditions and background clutter, which can introduce noise into the dataset. Therefore, it's essential to carefully manage these factors to maintain the integrity of our data.

By balancing the distribution of beans across regions and quality grades, we aim to prevent biases that could affect the performance of our deep-learning model. Incorporating beans from diverse sources and quality grades ensures that our model can effectively generalize to unseen coffee bean images, ultimately enabling accurate classification and analysis.

Table 3.1 No of coffee bean collected data

No	Origin	Processing type	Grade Level	No of Bean
1	Arsi	Unwashed	Grade 1	142
			Grade 2	163
		Washed	Grade 1	156
			Grade 2	187
2	YirgaCheffe	Unwashed	Grade 1	220
			Grade 2	160
			Grade 3	221
		Washed	Grade 1	272
			Grade 2	257
			Grade 3	258
3	Gujii	Unwashed	Grade 1	200
			Grade 2	194
			Grade 3	201
		Washed	Grade 1	188
			Grade 2	118
			Grade 3	180
4	Sidama	Unwashed	Grade 1	171
			Grade 2	176
		Washed	Grade 1	171
			Grade 2	178
			Grade 3	152
		Total	**21 Classes**	**3965**

3.2.1 Image Acquisition

Image acquisition refers to the process of capturing digital images using various devices such as cameras, scanners, or sensors. Image acquisition plays a critical role in generating high-quality datasets that are essential for training deep learning models to perform effectively in real-world applications. By ensuring that the acquired images are representative of the target domain and exhibit the necessary characteristics for model training, image acquisition sets the foundation for the success of subsequent deep-learning tasks. During the image-capturing process, we utilized a Smartphone camera, specifically the Infinix Hot 10 model, which is equipped with a 13-megapixel camera. The images were taken at a standard aspect ratio of 4:3 and a resolution of 4160x3120 pixels. We collected a total of 3965 coffee beans from all four locations: Yirgacheffe, Sidama, Gujii, and Arisi. Each coffee bean was individually photographed, resulting in a total of 6373 captured images across all locations. When taking the pictures, we ensured to capture images of both the front and back faces of the coffee beans. This approach allowed us to create a comprehensive dataset that includes images from different angles, providing a more detailed view of each bean's characteristics.

Despite some challenges, such as variations in lighting and focus, we adjusted camera settings and angles to ensure consistent image quality. Additionally, we employed post-processing techniques to enhance the images and make them suitable for further analysis. Overall; the image-capturing process involved using the Infinix Hot 10 Smartphone camera to take multiple pictures of coffee beans from all four locations, resulting in a comprehensive dataset of 6373 images for further analysis and classification.

Table 3.2 No of coffee bean captured images

No	**Origin**	**Processing type**	**Grade Level**	**No of images**
1	Arsi	Unwashed	Grade 1	275
			Grade 2	303
		washed	Grade 1	296
			Grade 2	320
2			Grade 1	333

	YirgaCheffe	Unwashed	Grade 2	260
			Grade 3	317
		washed	Grade 1	335
			Grade 2	300
			Grade 3	447
3	Gujii	Unwashed	Grade 1	250
			Grade 2	327
			Grade 3	358
		washed	Grade 1	301
			Grade 2	200
			Grade 3	294
4	Sidama	Unwashed	Grade 1	309
			Grade 2	318
		washed	Grade 1	297
			Grade 2	281
			Grade 3	252
		Total	**21 classes**	**6373**

Figure3. 1 Cheffe washed grade one coffee bean original

3.3 Pre-processing

Preprocessing in the context of deep learning refers to the series of steps and techniques applied to raw data before it is fed into a deep learning model for training or analysis. The goal of preprocessing is to prepare the data in a format that is suitable for the model to effectively learn patterns and make accurate predictions. Preprocessing involves cleaning, transforming, and organizing the data so that it is in a form that the model can understand and learn from. In the context of processing coffee bean images, preprocessing involves several key steps:

3.3.1 Renamed

The renaming process involves assigning new names to the coffee bean images based on their origin, processing type, and grade level. For example, a coffee bean image from location Arsi_washed_or unwashed_grade1, 2, or3, might be named "arsi_w_g1_001.jpg".

To achieve this renaming in Python, we can use the **os** module, which provides functions for interacting with the operating system. Specifically, we'll use the **os. listdir()** function to iterate through the files in a directory, and the **os. rename()** function to rename each file. Steps of the renaming process using Python:

i. Define a function to rename the images.
ii. Iterate through each image in the specified folder.

iii. Extract relevant information from the image filename, such as origin, processing type, and grade level.
iv. Construct a new filename based on the extracted information.
v. Use the **os. rename()** function to rename the image with the new filename.

3.3.2 Resized

Resizing images is a common preprocessing step in deep learning and computer vision tasks. It involves changing the dimensions of the images while preserving their aspect ratio. Resizing is often done to standardize the size of images in a dataset, making them uniform and compatible with the input requirements of CNN learning models. In our case using Python library used for image resizing is PIL (Python Imaging Library), which is now maintained as Pillow. Pillow provides a simple and effective way to resize images with various interpolation methods, ensuring high-quality results. Resizing images from 4160x3120 to 224x224 involves reducing the dimensions while preserving the aspect ratio. We can achieve this using the resize () method provided by the Pillow library. Steps of resizing:-

i. We import the **Image** module from the **PIL** library to work with images.
ii. The **resize_images()** function takes the folder path containing the images, as well as the new width and height for resizing.
iii. We iterate through each image file in the specified folder.
iv. For each image, we open it using **Image.open()**, resize it to the specified dimensions using the **resize()** method, and then save the resized image back to the original file location using **save()**.

Figure3. 2 Pre-processed cheffe_washed_grade_one coffee bean image [renamed, and resized]

3.3.3 Balancing the Dataset

Balanced refers to achieving an even distribution or representation across different categories or classes within a dataset. In the context of deep learning, balancing the dataset involves ensuring that each class or category has a similar number of instances, preventing biases and allowing the model to learn effectively from all classes. Balancing data is a crucial step in machine learning, especially when dealing with imbalanced datasets where some classes are underrepresented compared to others. In our case, we used oversampling. Oversampling involves increasing the number of instances in the minority class to match the number of instances in the majority class. This can be done by duplicating existing minority class instances or by generating new instances through techniques like data augmentation. In our case, where coffee bean images from the minority class are rotated and saved, this is an example of oversampling through data augmentation.

i. Load randomly the original coffee bean image from the minority class.
ii. Transform the image via rotation i.e. rotates the image by 90 degrees.
iii. Take the transformed image and write it back to the class.
iv. Repeat 1, 2, and 3 for a total of N times where N refers to the total number of augmented images.

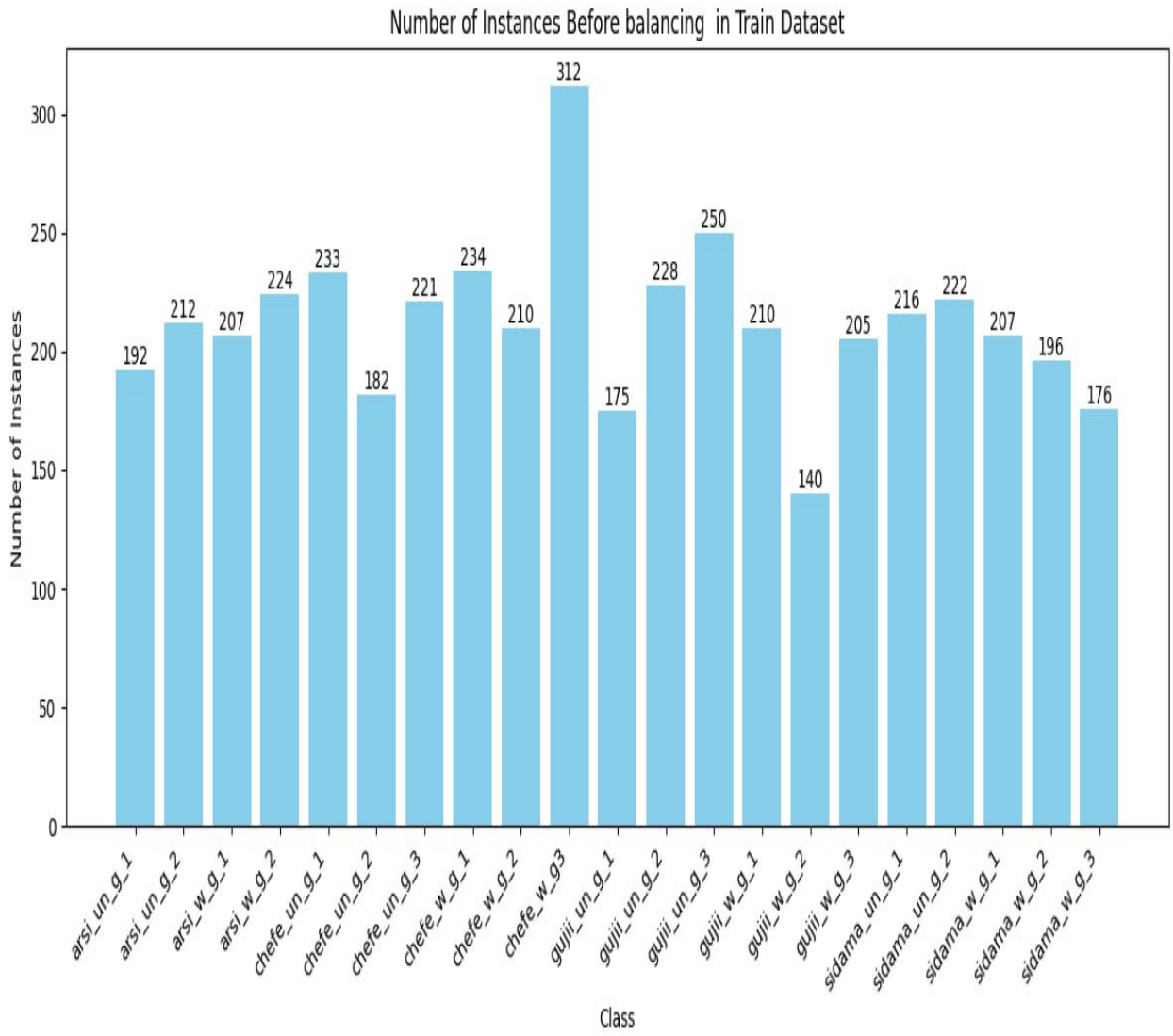


Figure 3 3 Bar chart before balancing the Coffee bean image of each class

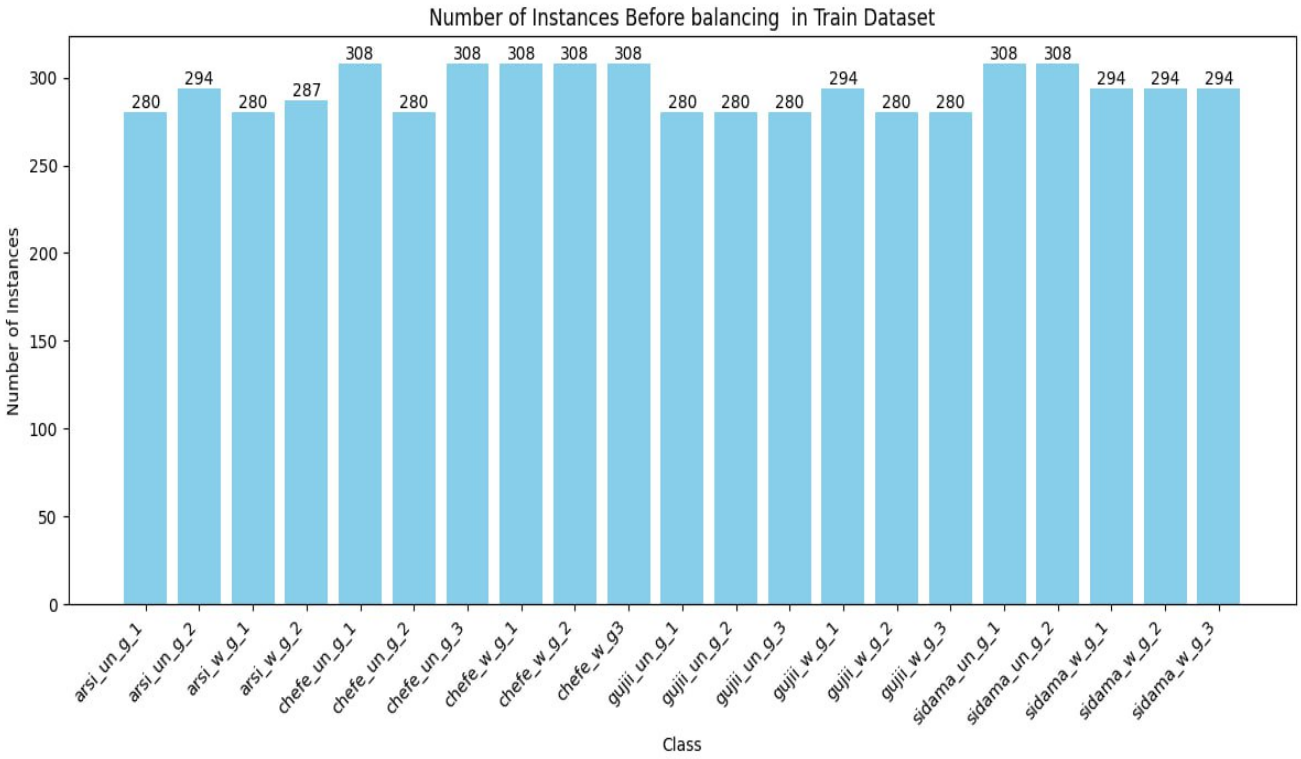


Figure3. 4 Bar Chart after balancing the Coffee bean image of each class

3.3.4 Normalizing Image Data

Normalizing image data is a crucial preprocessing step in deep learning and computer vision tasks. Normalization involves scaling the pixel values of images to a standard range, typically between 0 and 1 or -1 and 1. This process ensures that all features (pixel values) have a similar scale, which helps the model converge faster during training and improves its ability to learn meaningful patterns. Scale the pixel values of the image arrays to a desired range. The most common approach is to divide the pixel values by the maximum pixel value (e.g., 255 for images with 8-bit color depth) to normalize them to the range [0, 1].

Steps of Normalizing Image Data

i. We define a function **normalize image ()** that takes an image as input and returns the normalized image.
ii. Inside the function, we convert the image to a numpy array using **np. array()**.
iii. We then divide each pixel value by 255.0 to normalize it to the range [0, 1].
iv. Finally, we return the normalized image.

3.3.5 Splitting Image Data

Splitting image data within the context of deep learning involves dividing a dataset of images into subsets for training, validation, and testing. This process ensures that deep learning models are trained on a diverse range of examples, validated on a separate set to tune their parameters, and tested on yet another independent set to evaluate their performance accurately. Python libraries commonly used for splitting image data include **scikit-learn**, TensorFlow, and PyTorch. These libraries provide functions and utilities to facilitate the partitioning of datasets based on specified ratios or strategies, ensuring robust model training and evaluation. Steps

i. We define a function **split_data()** that takes the path to the image folder as input and returns three lists containing the filenames for the training, validation, and testing sets.
ii. We shuffle the list of image filenames to ensure randomness in the split.
iii. We define the split ratios for training, validation, and testing commonly used as 70:15:15 or 80:10:10
iv. We calculate the number of samples for each subset based on the defined ratios and the total number of images.
v. We perform the splitting using the **np. split()** function from NumPy.
vi. Finally, we return the lists containing the filenames for the training, validation, and testing sets.

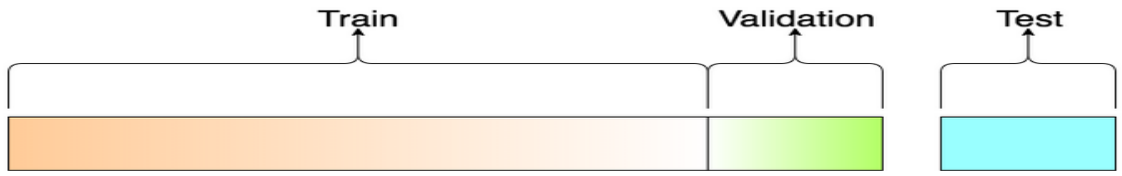


Figure3. 5 Visualization of Coffee bean image dataset split

3.4 System Architecture

The proposed system begins by capturing images of coffee beans using an image acquisition component. These images undergo pre-processing to enhance their quality. The improved images are then divided into three sets: one for training the proposed CNN model, another for validating the model's performance during training, and the third for testing the trained model's performance. The CNN model is trained using the training dataset. During training, the model's performance is regularly assessed using the validation dataset to ensure its learning effectively. Once training is

complete, the model is evaluated using a separate testing dataset. Throughout this iterative process, each stage of developing and refining the techniques and algorithms is thoroughly discussed. Additionally, the results achieved and the challenges encountered are carefully analyzed and explained.

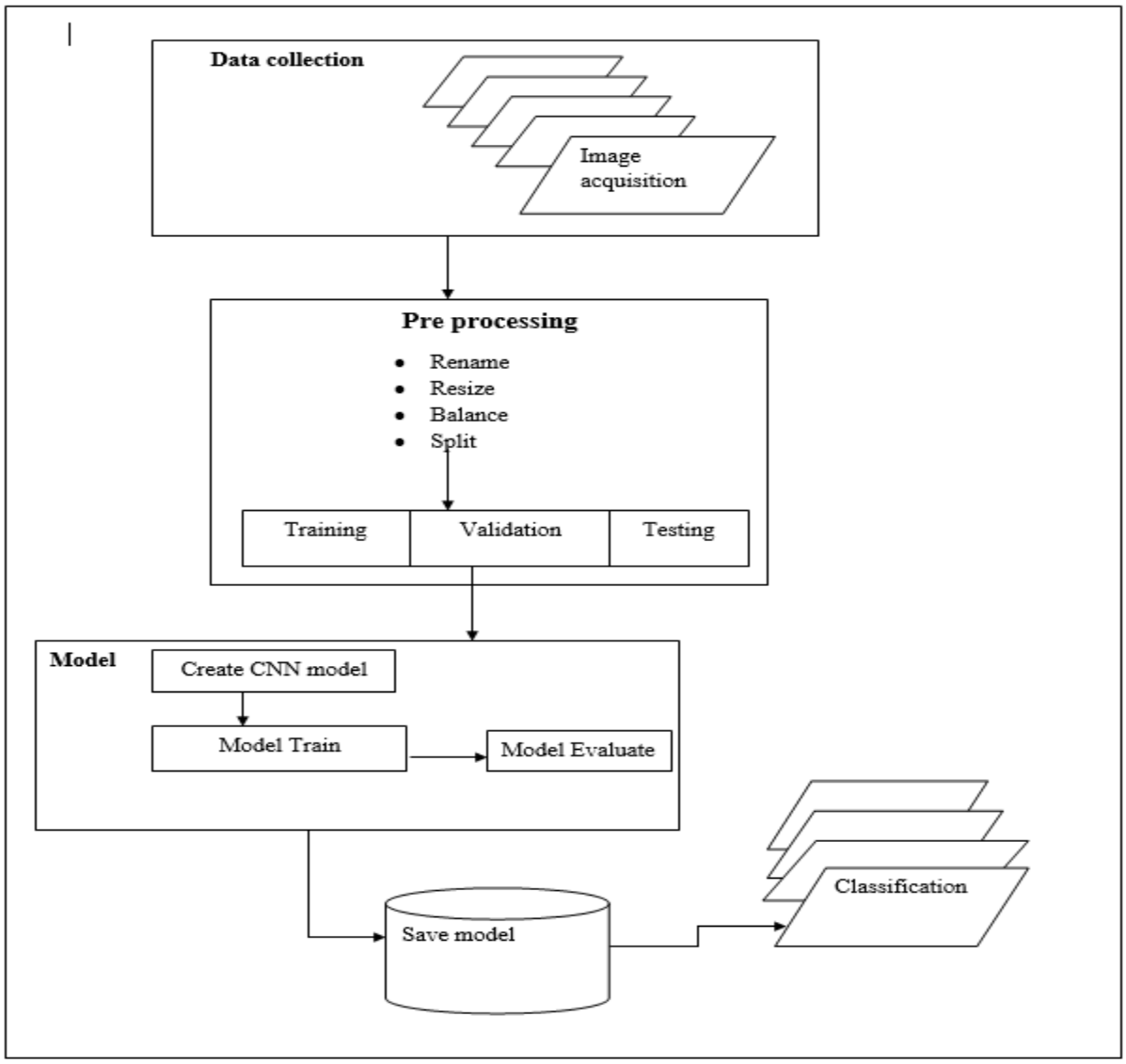


Figure3. 6 Block diagram of the proposed system architecture

3.5 Convolutional Neural Networks (CNNs)

Convolutional Neural Networks (CNNs) are a class of deep neural networks specifically designed for processing structured grid data, such as images. Convolutional Neural Networks (CNNs) are a powerful and versatile tool for image-based deep learning tasks. Their ability to automatically

learn spatial hierarchies of features from raw pixel data, along with their efficiency, scalability, and availability of pre-trained models, makes them a compelling choice for training model selection in various computer vision applications.

In this research, we focus on Convolutional Neural Networks (CNNs), a specialized class of deep neural networks tailored for processing structured grid data, particularly images. CNNs automatically learn hierarchical representations of features directly from raw pixel inputs. They comprise essential components such as convolutional layers, responsible for extracting local features using learnable filters, and pooling layers, which downsample feature maps to enhance translation invariance. Activation functions like ReLU introduce non-linearity, while fully connected layers aid in classification or regression tasks. Regularization techniques such as dropout and optimization algorithms or Adam are employed to optimize model training. Leveraging pre-trained models like VGG or ResNet offers transfer learning capabilities, allowing the utilization of knowledge learned from large-scale datasets such as ImageNet. By focusing on these CNN components, our research aims to achieve robust and efficient performance in image-based tasks.

3.5.1 The Proposed Model

A Convolutional Neural Network (CNN) has several layers, including an input layer where data centers and an output layer where results come out. It also has hidden layers in between. These hidden layers consist of convolutional layers, which use a technique called convolution to process the data. After convolution, there's an activation function that adds non-linearity to the data. Then, there might be more convolution or pooling layers. Fully connected layers are also hidden layers, as they're influenced by the activation function. Our CNN model includes an input layer, four convolutional layers, a layer that flattens the data, two fully connected layers, and an output layer. Each convolutional layer is followed by an activation and pooling layer. This **figure** helps the network learn and make sense of the input data.

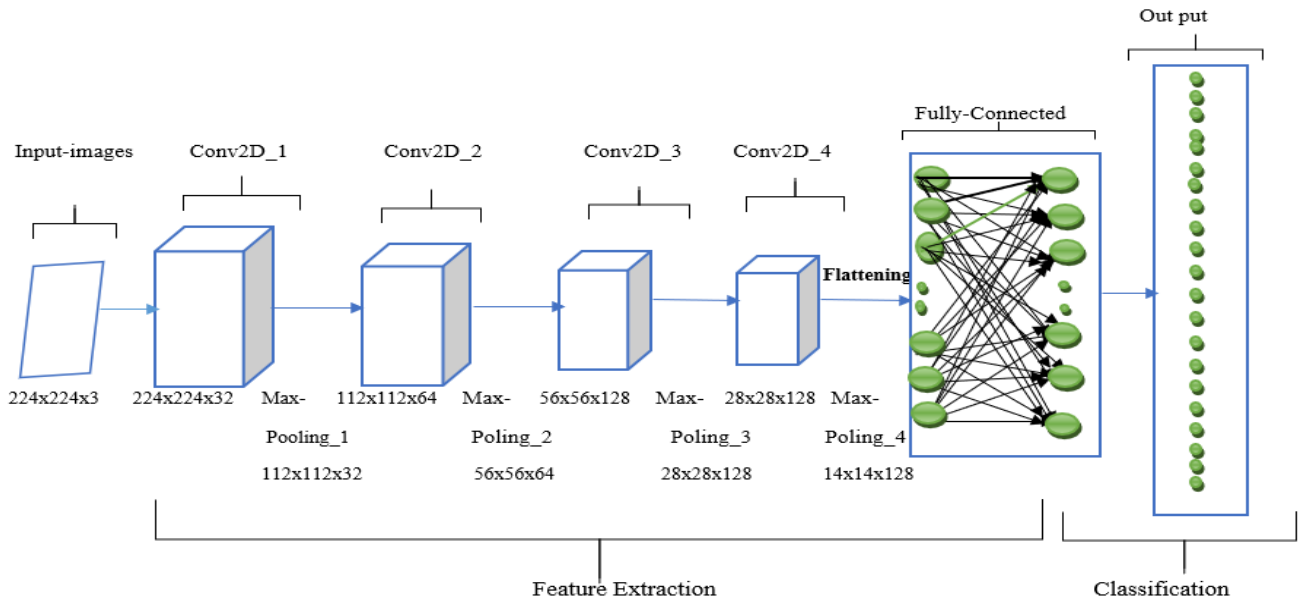


Figure3. 7 CNN proposed Model architecture

The proposed model was a Convolutional Neural Network (CNN) designed for image classification tasks. It consists of several convolutional layers, pooling layers, fully connected layers, and dropout layers. Below is a detailed description of each layer, including the calculation of the parameter values. The convolutional neural network (CNN) designed for multi-label image classification is composed of multiple layers, each playing a significant role in processing the input data, extracting features, reducing dimensionality, and classifying the images into multiple labels. This detailed description of each layer, along with parameter calculations, elucidates the complexity and capacity of the model. The input layer of the model is designed to accept images with a specific shape of 224x224 pixels and three color channels (RGB). This configuration ensures that each input image is represented as a three-dimensional matrix of size 224x224x3. The primary function of the input layer is to define the shape of the input data that will be processed by the subsequent layers of the neural network. It does not contain any trainable parameters. The chosen image size balances the need for a sufficient resolution to capture important features with computational efficiency. The choice of 224x224x3 is a common practice in many image classification tasks because it provides a good balance between resolution and computational efficiency. Images of this size are detailed enough to capture significant features necessary for accurate classification while being small enough to be processed efficiently by the network.

The first convolutional layer in the model employs a 3x3 filter size with 32 filters and applies the same padding to maintain spatial dimensions. ReLU activation is applied to introduce non-linearity, Max pooling with a pool size of 2x2 is performed to down-sample the feature maps. The Conv2D layer was a crucial component of Convolutional Neural Networks (CNNs). It performs convolutions, a fundamental operation where a set of filters (kernels) slides over the input image, computing the dot product between the filter values and the input values. This operation allows the layer to detect various features such as edges, textures, and patterns, which are essential for image recognition tasks. Let's see how to calculate the Parameter values. In this model, the filter size is 3×33×3. A stride of 1 is used, meaning the filter moves one pixel at a time. Padding is likely used to maintain the spatial dimensions.

$$\text{Output Dimensions} = \left(\frac{W-F+2P}{S}+1, \frac{H-F+2\text{P}}{S}+1, D\right) \qquad \text{Eq (3.1)}$$

$$\text{Number of Parameters} = (\text{F}\times\text{F}\times\text{Input Channels}+1)\times\text{D} \qquad \text{Eq (3.2)}$$

Where W and H are the input width and height, F is the filter size, P is the padding, S is the stride, and D is the number of filters.

The parameters and dimensions of this layer are calculated by using Eq 3.1 and Eq3.2 formulas for the given values W=224, H=224, P=1, F=3, S=1 and D=32

$$\text{Output Width} = \left(\frac{224-3+2x1}{1}+1\right) = 224$$

$$\text{Output Height} = \left(\frac{224-3+2x1}{1}+1\right) = 224$$

So, the output dimensions of the first layer are = 224x224x32.

The number of Parameters in the first convolution layer is = (3×3×3+1)×32=896.The parameters for this layer total 896, calculated from the weights and biases associated with the filters. Additionally, each filter has one bias term, resulting in a total of 896 parameters. After this layer apply the first max-pooling layer. Max-pooling is a down-sampling operation commonly used in CNNs to reduce the spatial dimensions of the feature maps and to make the representations more manageable computationally. This layer applies a 2x2 pooling window with a stride of 2, meaning that it moves two pixels at a time across the feature map. The 'Valid' padding indicates that no

padding is added to the input before pooling. In max-pooling, the maximum value within each window is taken as the output, which helps in retaining the most prominent features while discarding less significant information. This not only reduces the computational load but also makes the model more invariant to small translations in the input image. Since max-pooling involves a simple operation without any learnable parameters, this layer doesn't contribute any additional parameters to the model.

Output Dimensions of max-pooling = $\left(\frac{W}{S}, \frac{H}{S}, D\right)$ Eq (3.3)

Input size: 224x224x32

Pool size: 2x2

Stride: 2

The max pooling doesn't change the depth (number of channels), only the spatial dimensions are affected. Using max pooling with a pool size of 2x2 and stride of 2:

Output Dimensions of max-pooling =$\left(\frac{224}{2}, \frac{224}{2}, 32\right)$

So, the output dimensions of the first max-pooling layer are 112x112x32.

The second convolutional layer utilizes a 3x3 filter size with 64 filters. Similar to the first layer, applies the same padding. ReLU activation and Max pooling with a pool size of 2x2 are performed to down-sample the feature maps. This Conv2D layer applies more filters that already extracted features from the previous Conv2D one and Max-Pooling one layer. It aims to capture more complex patterns by increasing the depth of the feature maps. This deeper convolution layer can detect intricate details by combining the basic features identified earlier. This Conv2D layer is similar to those of the first but with a higher number of filters. The layer has 64 filters, allowing it to capture a wider range of patterns and details. The network learns hierarchical feature representations, where earlier layers capture simple features and later layers capture complex combinations. The filters in Conv2D layer two share weights across the entire input, significantly reducing the number of parameters compared to fully connected layers and making the model more efficient. The number of Parameters in the Convolutional Layer two $(3\times3\times32+1)\times64=18,496$

based on the Eq(3.2) formulas and we got the output dimensions of the second max-pooling layer are 56x56x64 in the Eq(3.3) formulas.

The third Conv2D layer applies 128 filters of size 3×3. This layer is crucial for identifying more sophisticated features, such as parts of objects and more detailed textures, by combining lower-level features from previous layers. The ReLU activation is applied after convolution, Max pooling with a pool size of 2x2 is performed to down-sample the feature maps. The convolutional layers continue to build a spatial hierarchy of features, where each layer's filters are sensitive to increasingly complex aspects of the input. The number of parameters in this convolutional layer is based on Eq(3.2) $(3\times3\times64+1)\times128=73,856$. The MaxPooling2D layer performs the third round of max pooling, further reducing the spatial dimensions of the input. This step is essential for progressively condensing the information and making the network more computationally efficient. we go it the output dimensions of the third max-pooling layer are 28x28x128 in the Eq(3.3) equation.

The fourth Conv2D layer continues the convolution process, applying another set of 128 filters with a 3x3 filter size to the input. This layer refines the extracted features further, capturing even more detailed patterns and spatial relationships. Similar to the previous layers, the kernel's same padding is applied. ReLU activation, finally, max pooling with a pool size of 2x2 is performed to down-sample the feature maps. This layer continues the hierarchical learning process, ensuring that the network can capture complex and abstract patterns necessary for accurate classification. The number of parameters in this convolutional layer is based on the Eq3.2 $(3\times3\times128+1)\times128=147,584$. The final MaxPooling2D layer performs the last stage of spatial down-sampling, reducing the feature map size to 14x14x128. This layer ensures that the features are compact and ready for the fully connected layers. After this layer is connected into one array mode by using the Flatten layer. The flattened layer converts the 3D feature maps into a 1D vector. This transformation is necessary for transitioning from the convolutional layers to the fully connected layers, which require a flat input. Transforms the 14x14x128 input into a single 1D vector of 25088 elements, preparing it for the dense layers. The flattened layer has no trainable parameters, as its sole function is to reshape the data.

The first dense layer is a fully connected layer with 512 units. Each of the 25088 features from the flattening layer is connected to each of the 512 units, resulting in a total of 25088 * 512 weights. Additionally, each unit has a bias term, leading to a total of 512 bias terms. This brings the total number of parameters in this layer to 12,845,568. Dense layers are used to perform the classification based on the features extracted by the convolutional layers. The ReLU activation function introduces non-linearity, allowing the model to learn complex patterns and relationships between the features.

$$\text{Number of Parameters} = (\text{Input Units} \times \text{Output Units}) + \text{Output Units} \quad \text{Eq(3.4)}$$

Number of Parameters in the first dense layer $=(25088 \times 512)+512=12,845,568$. This fully connected layer allows the model to learn complex combinations of the extracted features.

The first Dropout layer is a regularization technique that randomly sets a fraction of input units to zero during training. This helps prevent over-fitting by ensuring that the network does not rely too heavily on any single neuron. Dropout acts as a form of regularization, improving the network's ability to generalize to new, unseen data by reducing reliance on specific pathways. It randomly drops 40% of the units in the previous layer during training, helping to prevent over-fitting.

The second fully connected layer (Dense_2) with 256 neurons continues to process the features learned by the previous layers. This layer refines the learned features further and prepares the final representation for the output layer. Each of the 512 neurons from the previous layer is connected to each of the 256 neurons in this layer and Uses an activation function like ReLU to introduce non-linearity. This layer also has a significant number of parameters, reflecting its dense connections. Number of Parameters$=(512 \times 256)+256=131,328$ in the Eq3.4 formula. This layer further condenses the learned features into a more compact representation. The second dropout layer with a dropout rate of 0.2 follows, randomly dropping 20% of the units during training to further prevent over-fitting, without any learnable parameters. The final output layer is a dense layer with 21 units, corresponding to the 21 classes, and uses the sigmoid activation function, which is suitable for multi-label classification as it outputs a probability for each class independently.

$$\text{Number of Parameters} = (\text{Number of Input Units}+1) \times \text{Number of Output Units} \quad \text{Eq(3.5)}$$

The input shape to this layer is =256, the Number of Output Units =21, and the number of parameters is calculated as (256+1) × 21=5,397 parameters. These convolutional layers progressively extract and enhance features from the input data, contributing to the overall performance and effectiveness of the neural network model.

In general, we design the deep neural network architecture consisting of four Convolutional (Conv) layers with filter sizes of 32, 64, 128, and 128, respectively. Subsequently, three densely connected layers with sizes of 256 and 512 are added, followed by a final dense layer with a size of 21, representing the number of classes in the dataset. Summing up the parameters calculated in each layer, the network has a total of 13,223,125 parameters. All parameters in the Convolutional layers, pooling layers, and Fully Connected (FC) layers are trainable. Specifically, the network consists of 13,223,125 trainable parameters and 0 non-trainable parameters. Refer to the model summary provided see you below in Table 3.3 for a detailed overview. In this architecture, the total trainable parameters are 13,223,125, while the non-trainable parameters remain at 0.

Table 3 3 proposed model summary

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 224, 224, 32)	896
max_pooling2d (MaxPooling2D)	(None, 112, 112, 32)	0
conv2d_1 (Conv2D)	(None, 112, 112, 64)	18,496
max_pooling2d_1 (MaxPooling2D)	(None, 56, 56, 64)	0
conv2d_2 (Conv2D)	(None, 56, 56, 128)	73,856
max_pooling2d_2 (MaxPooling2D)	(None, 28, 28, 128)	0
conv2d_3 (Conv2D)	(None, 28, 28, 128)	147,584
max_pooling2d_3 (MaxPooling2D)	(None, 14, 14, 128)	0
flatten (Flatten)	(None, 25088)	0
dense (Dense)	(None, 512)	12,845,568
dropout (Dropout)	(None, 512)	0
dense_1 (Dense)	(None, 256)	131,328
dropout_1 (Dropout)	(None, 256)	0
dense_2 (Dense)	(None, 21)	5,397

3.5.2 Selecting the Optimizer

We have selected the Adam optimizer for our model. Adam is an adaptive optimization algorithm that combines the advantages of both AdaGrad and RMSProp. It's known for its effectiveness in a wide range of tasks due to its adaptive learning rates and momentum. By using Adam, our model can efficiently update its parameters during training, potentially leading to faster convergence and improved performance compared to other optimizers. We have a choice of 'binary_crossentropy' as the loss function indicates that we are dealing with a binary classification problem. This loss function is well-suited for tasks where the model is trained to output probabilities for two classes and penalizes deviations from the true binary labels. It measures the dissimilarity between the actual and predicted probability distributions, guiding the model to make more accurate predictions. Accuracy is the metric to monitor during training and evaluation. Accuracy measures the proportion of correctly classified samples out of the total number of samples. It provides a straightforward assessment of the model's performance in terms of classification accuracy, making it a commonly used metric for classification tasks.

3.5.3 Training the Model

Training a model in the context of machine learning and deep learning involves the process of iteratively adjusting the parameters of the model to minimize a predefined objective function, typically a loss function, on a training dataset. This process enables the model to learn patterns and relationships from the input data, which in turn allows it to make accurate predictions or classifications on new, unseen data. Training model a neural network involves both forward and backpropagation. The forward propagation, where input data is passed through the network to generate predictions, and backpropagation, where the error between predicted and actual outputs is propagated backward through the network to update its parameters.

Forward Propagation:

i. Input data (images) are fed into the network.
ii. Each layer in the network performs a linear transformation followed by a non-linear activation function.
iii. The output of each layer becomes the input to the next layer until the final output layer is reached.
iv. The final output layer produces predictions for each class.

Backpropagation:

i. Calculate the loss between the predicted output and the actual labels using a suitable loss function (e.g., binary cross-entropy for multi-label classification).
ii. Compute the gradient of the loss function concerning the parameters of the network using the chain rule.
iii. Update the parameters of the network using an optimization algorithm (e.g., Adam optimizer) to minimize the loss.
iv. Repeat steps 1-3 for a predefined number of epochs or until convergence.

3.5.4 Evaluating the Model

After training the model for a certain epoch on the training data, the performance of the model is evaluated on unseen data. In the pre-processing stage, the coffee bean images are divided into the training, validation, and test datasets. Now, we evaluate the model using the test dataset to determine how well the model performs. To measure the performance of the model, we have chosen different metrics such as classification accuracy, confusion matrix, precision, recall, and F1-score.

Classification accuracy is one of the primary metrics used to evaluate the performance of a model, especially in classification tasks. It measures the ratio of correctly predicted instances to the total instances in the dataset.

$$accuracy = \frac{nunber\ of\ correct\ prediction}{Total\ number\ of\ predction} \qquad \text{Eq (3.6)}$$

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Confusion Matrix: A confusion matrix is an essential tool in evaluating the performance of a classification model, providing a comprehensive view of how well the model's predictions match the actual ground truth labels. In the context of our multi-label classification model for coffee bean images, the confusion matrix gives insights into the model's accuracy, precision, recall, and the distribution of errors across different classes. Construction of the Confusion Matrix:

Matrix Layout The confusion matrix is a square matrix where each row represents the actual classes, and each column represents the predicted classes. For a multi-class classification problem with NxN classes, the confusion matrix will have dimensions N×N. Elements of the Matrix.

True Positives (TP): The number of times the model correctly predicted a class. These are found on the diagonal of the matrix.

False Positives (FP): The number of times the model incorrectly predicted a class. These are found in the columns but outside the diagonal.

True Negatives (TN): The number of times the model correctly predicted the absence of a class. These are implicitly included but not explicitly shown in the matrix.

False Negatives (FN): The number of times the model failed to predict a class. These are found in the rows but outside the diagonal.

Evaluating the performance of a classification model involves more than just calculating its overall accuracy. Precision, recall, and F1-score are crucial metrics that provide deeper insights into how well the model is performing, especially in multi-class classification scenarios. These metrics help to understand the balance between correctly identified instances and misclassified instances for each class. Here, we will discuss these metrics in detail, explaining their significance and how they are calculated.

Precision

Precision for a specific class i is defined as the ratio of true positive predictions (TP) to the sum of true positive and false positive predictions (FP) for that class. Mathematically, it is given by:

$$\text{precision} = \frac{TP\text{i}}{TPi+FPi} \qquad \text{Eq (3.7)}$$

High precision indicates that class i has few false positives, meaning that when the model predicts class i, it is usually correct. Precision is particularly important in scenarios where the cost of false positives is high. For instance, in our coffee bean classification model, high precision ensures that when a specific type of coffee bean is identified, it is very likely to be that type minimizing the risk of mislabeling.

Recall

Recall for a specific class i is defined as the ratio of true positive predictions (TP) to the sum of true positive and false negative predictions (FN) for that class. It is calculated as:

$$\text{Recall} = \frac{TP\text{i}}{TPi+FNi} \qquad \text{Eq (3.8)}$$

High recall indicates that class iii has few false negatives, meaning the model successfully identifies most instances of class iii. Recall is crucial in situations where missing true instances (false negatives) is particularly undesirable. In this case, the coffee bean classification model, high recall ensures that most coffee beans of a specific type are correctly identified, reducing the chances of overlooking any instances of that type.

F1-Score

The F1-score is the harmonic mean of precision and recall, providing a single metric that balances both. It is given by:

$$\text{F1} - \text{score} = \frac{2 \cdot \text{Precisioni} \cdot \text{Recalli}}{\text{Precisioni} + \text{Recalli}} \qquad \text{Eq (3.9)}$$

The F1 score is particularly useful when there is an uneven class distribution or when both precision and recall are important for the application. It ensures that a model does not achieve high performance in one metric at the expense of the other. For our coffee bean classification model, the F1-score provides a comprehensive measure of performance by balancing precision and recall, ensuring that the model is both accurate and reliable in identifying different types of coffee beans. Understanding and calculating these metrics for each class in the confusion matrix helps in diagnosing the strengths and weaknesses of the model. For example, if the model has high precision but low recall for a specific class, it means the model is conservative in making predictions for that class and may miss many true instances. Conversely, high recall but low precision indicates that the model is overzealous in predicting that class, resulting in many false positives.

CHAPTER FOUR

4. Demonstration and Evaluation

4.1. Overview

The main objective of this chapter is to demonstrate and evaluate different CNN architectures on different dataset sizes. The chapter is subdivided into three sub-sections. In the first subsection, the designed model will be demonstrated through different experiments. At the end of each experiment, the performance of the model is evaluated using different evaluation metrics discussed in chapter three. The evaluation summary of the different models is covered in the second subsection. The experiment results are also compared in this subsection.

4.2. Demonstration of the Experiments

We designed the model in Chapter Three. According to Puffer's DSR guidelines, the fourth step involves demonstrating the artifact. In the following section, we will demonstrate the model through various experiments. Before detailing each experiment, let's explain how we divided the coffee bean images into training, validation, and test datasets. We used two different methods for splitting the images. In the first method, 70% of the images from each coffee bean class were used for training, 15% for validation, and the remaining 15% for testing.

In the second method, 80% of the images from each coffee bean class were used for training, 10% for validation, and 10% for testing. Note that these percentages apply to each coffee bean class individually. Table 4.1 shows the number of images used for training, validation, and testing the model for each coffee bean class in both methods.

Table 4. 1 number of images used for training, validation, and testing the model from each

Coffee bean classes	**The number of coffee bean images split Datasets**					
	Dataset 70-15-15			**Dataset 80-10-10**		
	Train	**Validation**	**Test**	**Train**	**Validation**	**Test**
Arsi_unwashed_Grade1	280	60	60	320	40	40
Arsi_unwashed_Grade2	294	63	63	336	42	42

Arsi_washed_Grade1	280	60	60	320	40	40
Arsi_washed_Grade2	287	61	62	328	41	41
Chefe_unwashed_Grade1	308	66	66	352	44	44
Chefe_unwashed_Grade2	280	60	60	320	40	40
Chefe_unwashed_Grade3	308	66	66	352	44	44
Chefe_washed_Grade1	308	66	66	352	44	44
Chefe_washed_Grade2	308	66	66	352	44	44
Chefe_washed_Grade3	308	66	66	352	44	44
Gujii_unwashed_Grade1	280	60	60	320	40	40
Gujii_unwashed_Grade2	280	60	60	320	40	40
Gujii_unwashed_Grade3	280	60	60	320	40	40
Gujii_washed_Grade1	294	63	63	336	42	42
Gujii_washed_Grade2	280	60	60	320	40	40
Gujii_washed_Grade3	280	60	60	320	40	40
Sidama_unwashed_Grade1	308	66	66	352	44	44
Sidama_unwashed_Grade2	308	66	66	352	44	44
Sidama_washed_Grade1	294	63	63	336	42	42
Sidama_washed_Grade2	294	63	63	336	42	42
Sidama_washed_Grade3	294	63	63	336	42	42
Total	**6153**	**1318**	**1319**	**7032**	**879**	**879**

4.2.1 Experiment on Grayscale Images (Experiment 1 and 2)

The original coffee bean images we captured are in color. For this experiment, we converted these images to grayscale to reduce the training time. The goal is to assess how the color of the coffee beans affects the classification.

The CNN model consists of the following layers:

The CNN model has four convolutional layers having kernel size 32, 64, 128, and 128 for feature extraction followed by two densely connected layers of size 512, 256 and the output layer of size 21 (the number of classes of the coffee bean images). ReLU activation functions are used for the hidden layers and sigmoid activation functions are used for the output layer. We have also applied zero-padding for each Convolution layer followed by the max-pooling layer. In addition to this, we have added a dropout layer before flattening the feature maps and before the output layer with the dropout rate of 0.4 and 0.2 respectively. The main advantage of the dropout layer is that prevents the model from over-fitting.

Dataset

Coffee bean images that are converted to the grayscale format are used. We split the coffee bean images in two ways. In the first case, 70% of the dataset is used for training the model, 15% for validation, and 15% for testing the model. Whereas in the second case, 80% for training, 10 for validation, and 10 for testing the model.

Training the model

In both cases, the model undergoes compilation, a crucial step before the initiation of training. The Adam optimizer is employed for optimization, a popular choice due to its adaptive learning rate capabilities. In this implementation, the optimizer's learning rate is set to 0.001, ensuring gradual updates to model parameters during training to converge toward optimal values. Furthermore, two additional parameters, namely beta_1 and beta_2, govern the exponential decay rates for the first and second moments of the gradient, respectively. These parameters influence the algorithm's behavior, with beta_1 typically set to 0.9 and beta_2 set to 0.999 by default. These values contribute to the optimizer's ability to adaptively adjust learning rates and handle different types of data and optimization landscapes.

During the compilation phase, the model's loss function is specified as binary cross-entropy, a suitable choice for binary classification tasks. This function quantifies the difference between predicted probabilities and actual class labels, facilitating efficient training of models aimed at discerning between two classes. Additionally, accuracy is chosen as the evaluation metric, providing insights into the model's performance by measuring the proportion of correctly classified instances. Together, these components enable effective model evaluation and optimization, guiding the training process toward achieving desirable classification outcomes.

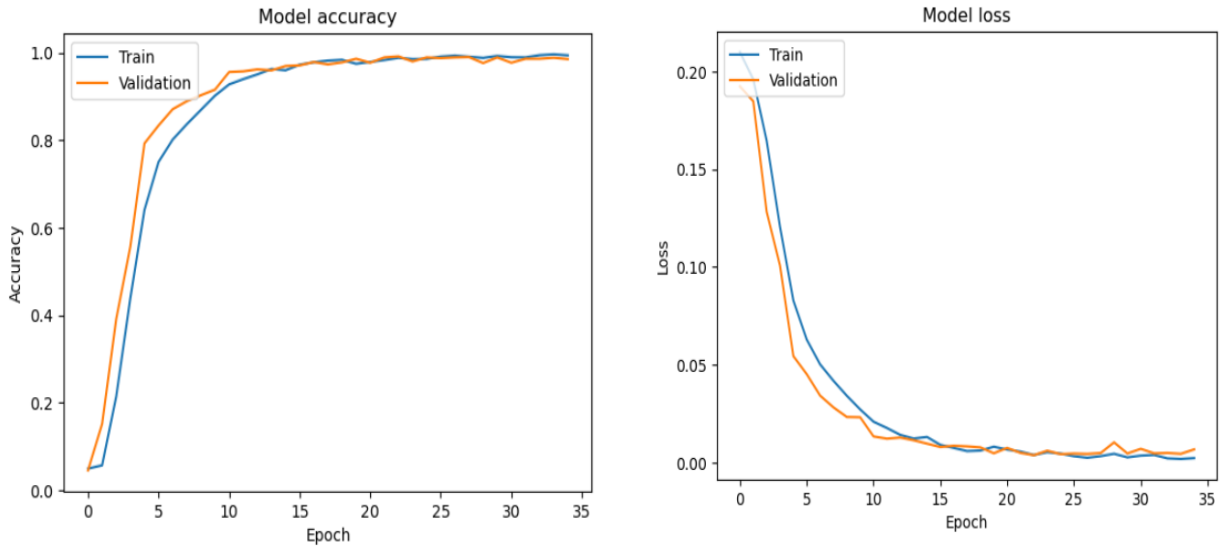


Figure 4.1 Training and validation of Grayscale dataset 70-15-15 (A) Model accuracy (B) model loss

The accuracy of the model on both the training and validation datasets increased significantly over the training epochs. The initial training accuracy was 4.38%, which gradually improved to 99.30% by the end of epoch 35. Similarly, the validation accuracy started at 4.55% and reached 98.56% by the end of the training process. The loss metric, which indicates the error in the model's predictions, decreased substantially over the epochs. The training loss started at 0.2418 and decreased to 0.0024 by epoch 35. The validation loss also showed a significant reduction, starting at 0.1923 and reaching 0.0068 at the end of the training.

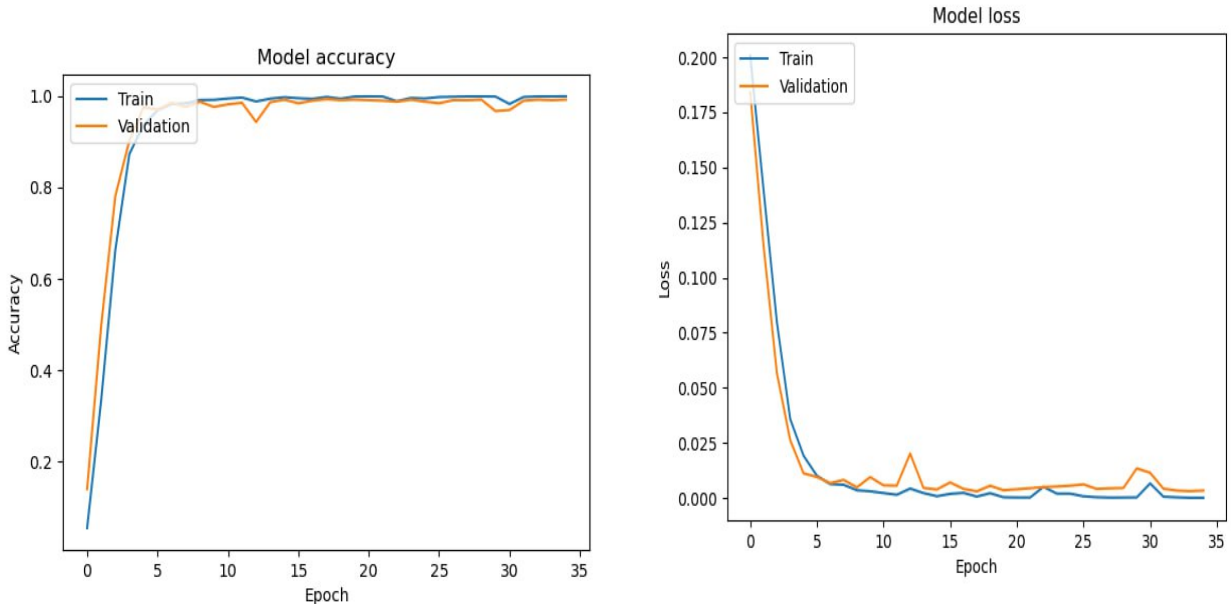


Figure 4. 2 Training and validation of Grayscale dataset 80-10-10 (A) Model accuracy (B) model loss

The model is continuously improving its accuracy and reducing its loss over time. The validation accuracy and loss also show a consistent improvement throughout the training process, indicating that the model is generalizing well to unseen data. The initial epochs (1-10) show a steady improvement in accuracy and a decrease in loss, with a significant increase in accuracy from 0.0466 to 0.8480. The validation accuracy also shows a consistent improvement, with a significant increase from 0.1388 to 0.9022.

The mid-training epochs (11-20) show a slight fluctuation in accuracy and loss, but overall, the model continues to improve its performance. The validation accuracy remains high, with a slight decrease in epoch 13, but recovers in subsequent epochs. The final epochs (21-35) show a significant improvement in accuracy, with a peak of 0.999 in epoch 35. The loss also decreases significantly, with a minimum value of 0.97161×10^{-5} in epoch 35. The validation accuracy also remains high, with a slight decrease in epoch 30, but recovers in subsequent epochs. The results demonstrate that the model is capable of learning from the training data and generalizing well to unseen data. The consistent improvement in accuracy and decrease in loss over time indicate that the model is learning effectively and adapting to the training data.

Evaluating the Model

The model is evaluated in both cases using the test dataset. The model scored 98.56% test accuracy and for the first test dataset whereas on the second dataset 99.09% test accuracy.

Case 1: Here, the model trained on 70% of the dataset is evaluated. We have got 98.56% accuracy on the test dataset

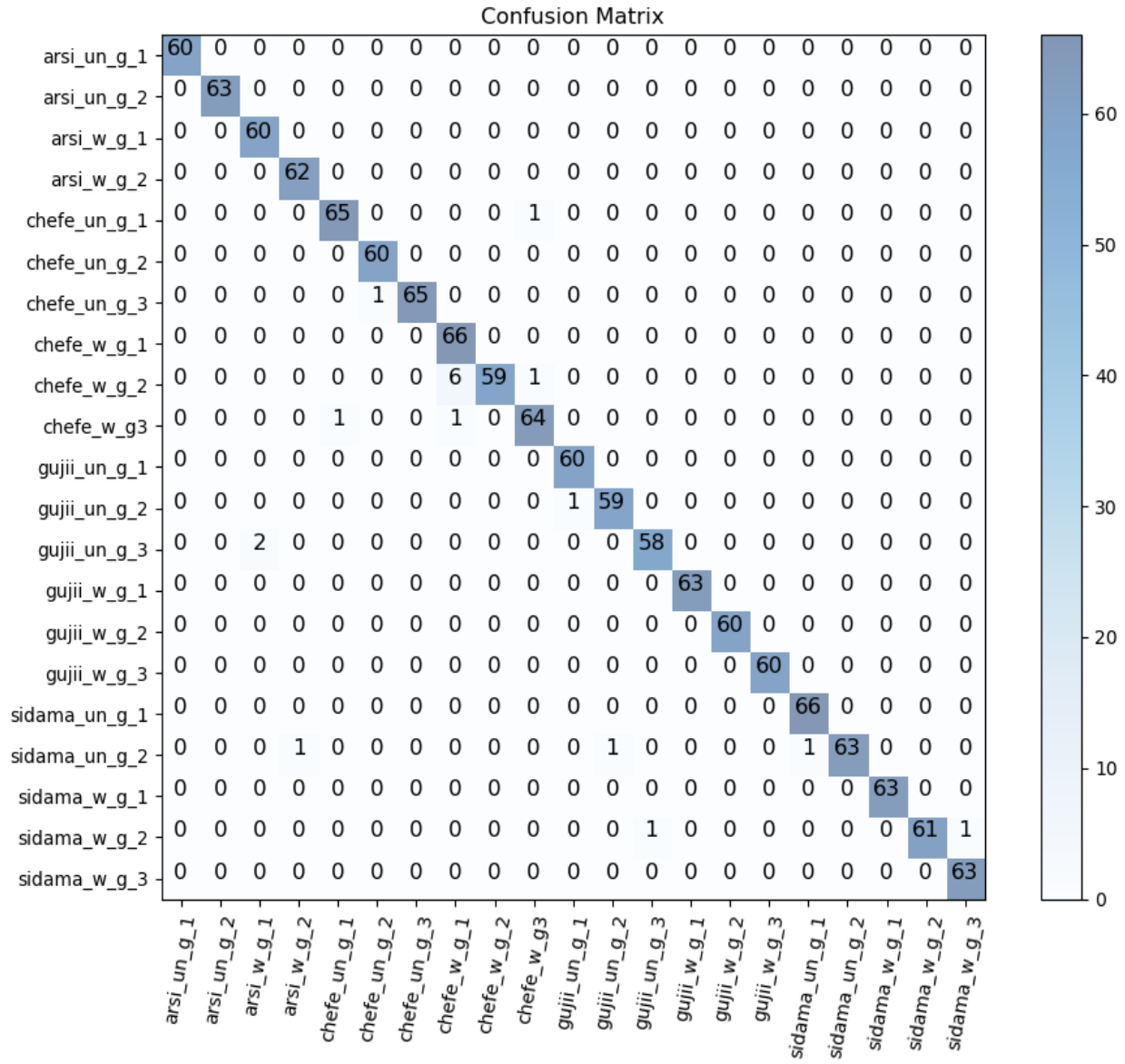


Figure 4.3 Confusion matrix of Grayscale dataset 70-15-15

The confusion matrix shows a comprehensive overview of the model's classification performance across 21 distinct classes, each representing a unique combination of location and quality. The matrix illustrates that the model achieved high accuracy, as evidenced by the majority of predictions being concentrated along the diagonal, indicating correct classifications. Each cell in the matrix represents the number of instances for which the actual class is the row and the predicted

class is the column, with diagonal elements denoting correct predictions and off-diagonal elements indicating misclassifications.

Analyzing the matrix in detail reveals that most classes exhibit very high correct classification rates. For example, the arsi_un_g_1 class has 60 correct predictions, chefe_w_g_1 has 66 correct predictions, and sidama_w_g_3 has 63 correct predictions out of their respective totals. This pattern of high accuracy is consistent across almost all classes, highlighting the model's strong performance. However, some misclassifications are present. For instance, the chefe_w_g_2 class has 59 correct predictions but seven instances misclassified with 6 instances classified as chefe_w_g_1 and another 1 as chefe_w_g_2. and the gujii_un_g_2 class has one instance misclassified as gujii_un_g_1. Notably, the gujii_un_g_3 class has 58 correct predictions but two misclassifications, with one instance classified as arsi_w_g_1.

	precision	recall	f1-score	support
arsi_un_g_1	1.00	1.00	1.00	60
arsi_un_g_2	1.00	1.00	1.00	63
arsi_w_g_1	0.97	1.00	0.98	60
arsi_w_g_2	0.98	1.00	0.99	62
chefe_un_g_1	0.98	0.98	0.98	66
chefe_un_g_2	0.98	1.00	0.99	60
chefe_un_g_3	1.00	0.98	0.99	66
chefe_w_g_1	0.90	1.00	0.95	66
chefe_w_g_2	1.00	0.89	0.94	66
chefe_w_g3	0.97	0.97	0.97	66
gujii_un_g_1	0.98	1.00	0.99	60
gujii_un_g_2	0.98	0.98	0.98	60
gujii_un_g_3	0.98	0.97	0.97	60
gujii_w_g_1	1.00	1.00	1.00	63
gujii_w_g_2	1.00	1.00	1.00	60
gujii_w_g_3	1.00	1.00	1.00	60
sidama_un_g_1	0.99	1.00	0.99	66
sidama_un_g_2	1.00	0.95	0.98	66
sidama_w_g_1	1.00	1.00	1.00	63
sidama_w_g_2	1.00	0.97	0.98	63
sidama_w_g_3	0.98	1.00	0.99	63
accuracy			0.99	1319
macro avg	0.99	0.99	0.99	1319
weighted avg	0.99	0.99	0.99	1319

Figure 4. 4 Classification Report of Grayscale Dataset 70-15-15

The classification report provides a detailed assessment of the model's performance across 21 classes, evaluating precision, recall, and f1-score, alongside support values for each class. The overall accuracy of the model was achieved at 99%, as reflected in the macro and weighted

averages, both standing at 0.99 for precision, recall, and f1-score. This high level of performance is consistent across nearly all classes.

The precision metric, which indicates the proportion of true positive predictions among the total predicted positives, is notably high for all classes, with most achieving a perfect score of 1.00. For instance, arsi_un_g_1, arsi_un_g_2, and chefe_un_g_3 all exhibit perfect precision. Similarly, gujii_w_g_1, gujii_w_g_2, gujii_w_g_3, sidama_un_g_2, sidama_w_g_1, and sidama_w_g_2 also achieve perfect precision, underscoring the model's ability to correctly identify instances of these classes without falsely classifying instances from other classes.

Recall, which measures the proportion of true positives among the actual positives, is equally impressive. Most classes report a recall of 1.00, indicating that the model effectively captures almost all instances of these classes. Minor deviations are observed in a few classes, such as chefe_w_g_2 and sidama_w_g_2, with recall values slightly below 1.00, suggesting a very small number of instances were missed.

The f1-score, a harmonic mean of precision and recall, further emphasizes the model's robust performance. The majority of classes report near-perfect or perfect f1-scores, reflecting the balance between precision and recall. For example, classes such as arsi_w_g_2, chefe_un_g_2, and gujii_un_g_1 have f1-scores of 0.99 or higher, indicating consistent performance across these metrics.

Support, which represents the number of true instances for each class, varies across classes but remains relatively balanced, ensuring that the performance metrics are not skewed by class imbalance. The model's ability to maintain high accuracy, precision, recall, and f1-scores across all classes demonstrates its robustness and effectiveness.

Case 2: Here, the model trained on 80% of the dataset is evaluated. We have got 99.09% accuracy on the test dataset

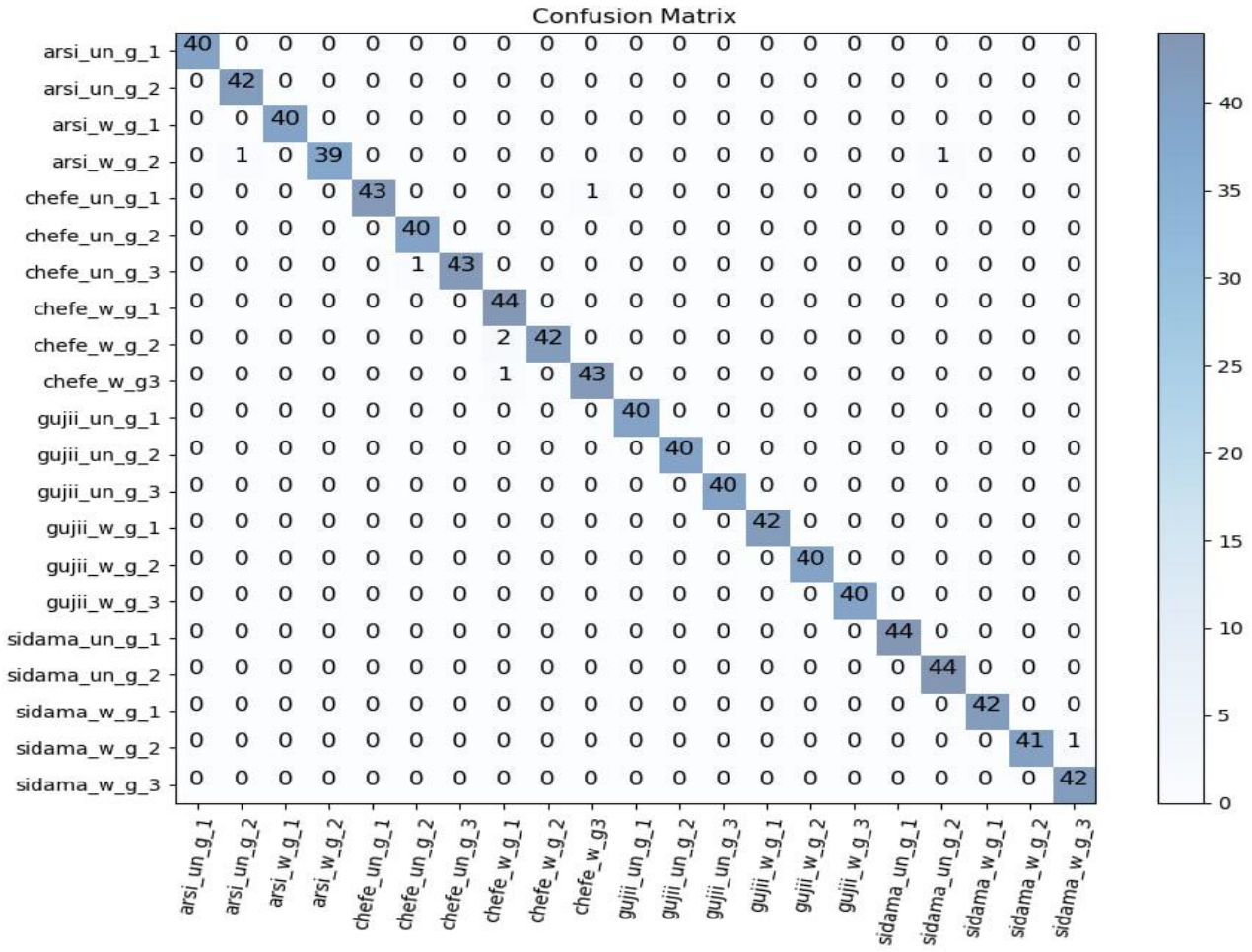


Figure 4.5 Confusion matrix of Grayscale dataset 80-10-10

The confusion matrix is a detailed representation of the model's performance across different classes. Each cell in the matrix indicates the number of predictions made by the model for a particular class (rows represent actual classes, and columns represent predicted classes). The evaluation focuses on highlighting instances of perfect classification, minor misclassifications, and areas for potential improvement. The evaluation encompasses a diverse set of class labels, each representing distinct categories within the dataset. Through meticulous analysis, this report provides insights into the model's precision, recall, and overall accuracy across different classes.

For the class label "arsi_un_g_1," the model achieves perfect classification, with no instances misclassified. All 40 samples are correctly identified, resulting in 40 true positives, and there are no false positives or false negatives. This indicates that the model distinguishes this class effectively. Similarly, "arsi_un_g_2" also shows no misclassifications. All 42 instances are

accurately predicted, leading to 42 true positives and zero false positives or false negatives. This perfect score underscores the model's precision for this class. The class "arsi_w_g_1" is another example of flawless classification, with all 40 samples correctly identified as true positives. There are no false positives or false negatives, indicating excellent model performance. In the case of "arsi_w_g_2," the model correctly classifies 39 out of 41 instances, with 2 samples misclassified as other classes, resulting in 39 true positives, 1 false positive, and 2 false negatives. Despite this, the model's accuracy remains high, indicating minor room for improvement.

For the class label "chefe_un_g_1," the model correctly identifies 43 out of 44 samples, with 1 sample misclassified, leading to 43 true positives and 1 false negative. This minor misclassification does not significantly impact the overall high accuracy. "Chefe_un_g_2" exhibits perfect classification with all 40 samples correctly identified, resulting in 40 true positives and no misclassifications. This shows the model's strong ability to recognize this class. Similar to other "chefe_un" classes, "chefe_un_g_3" shows high accuracy with 43 out of 44 samples correctly classified, resulting in 43 true positives and 1 false negative. The model performs well with minor misclassification.

For "chefe_w_g_1," all 44 samples are accurately predicted as true positives, indicating no false positives or false negatives and highlighting the model's perfect performance for this class. "Chefe_w_g_2" has 42 true positives out of 44 instances, with 2 samples misclassified as other classes, leading to 2 false negatives. Additionally, there are 2 false positives. Despite these errors, the overall accuracy remains high. The class "chefe_w_g_3" has 43 true positives out of 44 samples, with 1 misclassified instance resulting in 1 false positive and 1 false negative. This slight misclassification does not significantly detract from the high performance.

The classes "Gujii_un_g_1," "Gujii_un_g_2," and "Gujii_un_g_3" all three Gujii_un classes show perfect classification with 40 true positives each and no misclassifications, demonstrating the model's robust ability to identify these classes accurately. For "gujii_w_g_1," all 42 samples are correctly classified with no misclassifications, resulting in 42 true positives and highlighting the model's perfect performance. The classes "gujii_w_g_2" and "gujii_w_g_3" exhibit perfect classification with 40 true positives each and no false positives or false negatives, indicating flawless model performance for these classes.

The class labels Sidama_un_g_1, and Sidama_un_g_2 both "sidama_un" classes show perfect classification with 44 true positives each and no misclassifications, showcasing the model's excellent ability to identify these classes. "Sidama_w_g_1" also achieves perfect classification with 42 true positives and no misclassifications, reflecting the model's precise performance. For "sidama_w_g_2," the model correctly identifies 41 out of 42 samples, with 1 false negative and no false positives, indicating a high level of accuracy with slight room for improvement. Lastly, "sidama_w_g_3" achieves perfect classification with 42 true positives and no misclassifications, underscoring the model's strong performance.

Classification Report:

	precision	recall	f1-score	support
arsi_un_g_1	1.00	1.00	1.00	40
arsi_un_g_2	0.98	1.00	0.99	42
arsi_w_g_1	1.00	1.00	1.00	40
arsi_w_g_2	1.00	0.95	0.97	41
chefe_un_g_1	1.00	0.98	0.99	44
chefe_un_g_2	0.98	1.00	0.99	40
chefe_un_g_3	1.00	0.98	0.99	44
chefe_w_g_1	0.94	1.00	0.97	44
chefe_w_g_2	1.00	0.95	0.98	44
chefe_w_g3	0.98	0.98	0.98	44
gujii_un_g_1	1.00	1.00	1.00	40
gujii_un_g_2	1.00	1.00	1.00	40
gujii_un_g_3	1.00	1.00	1.00	40
gujii_w_g_1	1.00	1.00	1.00	42
gujii_w_g_2	1.00	1.00	1.00	40
gujii_w_g_3	1.00	1.00	1.00	40
sidama_un_g_1	1.00	1.00	1.00	44
sidama_un_g_2	0.98	1.00	0.99	44
sidama_w_g_1	1.00	1.00	1.00	42
sidama_w_g_2	1.00	0.98	0.99	42
sidama_w_g_3	0.98	1.00	0.99	42
accuracy			0.99	879
macro avg	0.99	0.99	0.99	879
weighted avg	0.99	0.99	0.99	879

Figure 4. 6 Classification report of Grayscale dataset 80-10-10

The classification report provides a comprehensive summary of the model's precision, recall, and F1 score for each class, along with overall accuracy. The precision and recall for most classes are either 1.00 or very close to it, indicating high model performance. A few classes have slightly lower scores (e.g., "arsi_w_g_2," has a recall of 0.95), but these deviations are minor. The F1 scores are similarly high, reflecting the balance between precision and recall for each class. The weighted and macro averages are both 0.99, which is an excellent result indicating overall high model performance across all classes. The support values show the number of samples per class, which are relatively balanced (ranging from 40 to 44 samples per class).

4.2.2. Experiments on RGB coffee bean images (Exp 3 and 4)

In this experiment, we train the model on the full-color images of the coffee beans images. The CNN Model has four convolutional layers for feature extraction and three fully connected layers for the classification of the coffee beans. The dropout regularization technique is also used.

CNN Architecture

The input layer accepts the images with 224x224x3.

i. Convolutional layer with 32 kernel size, 3x3 filters, followed by a max-pooling layer with 2x2 filters.
ii. Convolutional layer with 64 kernel size, 3x3 filters, followed by a max-pooling layer with 2x2 filters.
iii. Convolutional layer with 128 kernel size, 3x3 filters, followed by a max-pooling layer with 2x2 filters.
iv. Convolutional layer with 128 kernel size, 3x3 filters, followed by a max-pooling layer with 2x2 filters.
v. Flattening
vi. Two fully connected layers with 512 and 256 neurons followed by a dropout layer with a dropout rate of 0.4 (40%) and 0.2 (20%) respectively.
vii. Output layer with 21 neurons corresponding to the 21 coffee bean classes.

The ReLU activation function is used for the convolutional layer and the first two FC layers whereas the sigmoid activation function is used for the output layer. Zero-padding is applied for all convolutional layers.

Dataset

Coffee bean images with a size of 224x224x3 are used. We have used 70 % of the dataset for training, 15% for validation, and the remaining 15% for testing the model in the first experiment and 80% of the dataset for training, 10% for validation, and 10% for testing the model is used in the second experiment.

Training the model

We have trained the model in both cases for 35 epochs on the training data using the “Adam” optimizer with the learning rate of 0.001, beta_1=0.9, beta_2=0.999. A batch gradient descent with

a batch size of 32 is used that defines the amount of data the model samples at a single optimization or validation step. Since we are performing the multi-label classification, a loss function called binary_crossentropy is used. We have trained the model once again times and got the highest training accuracy of 99.06% and validation accuracy of 99.01% for the first dataset and 99.24% training accuracy and validation accuracy of 99.77% for the second dataset As we can see the model training progressed significantly over 35 epochs, beginning with an initial training accuracy of 5.98% and a validation accuracy of 19.04% in the first epoch. Rapid improvements were observed early, with training accuracy reaching 77.70% and validation accuracy at 86.04% by the third epoch. Consistent high performance followed, with training accuracy frequently exceeding 98% and validation accuracy steadily improving, surpassing 99% in several epochs. The best performance was noted in epoch 25, where training accuracy hit 99.79% and validation accuracy was 99.01%. The final epoch maintained these high standards with a training accuracy of 99.06% and a validation accuracy of 99.01%.

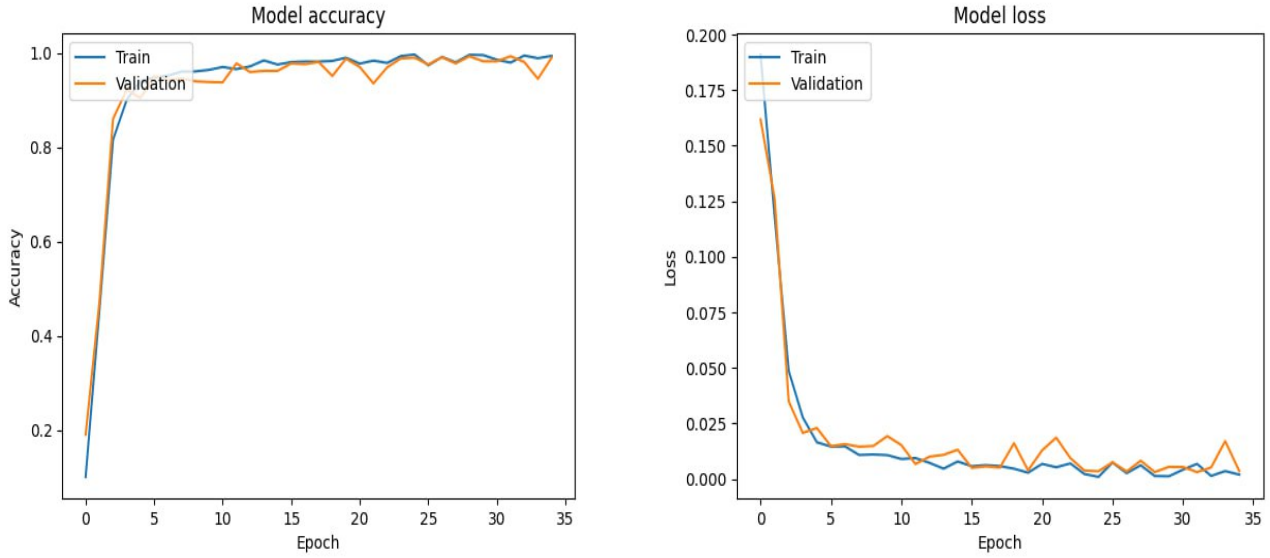


Figure 4.7 Training and validation of RGB dataset 70-15-15 (A) Model accuracy (B) model loss

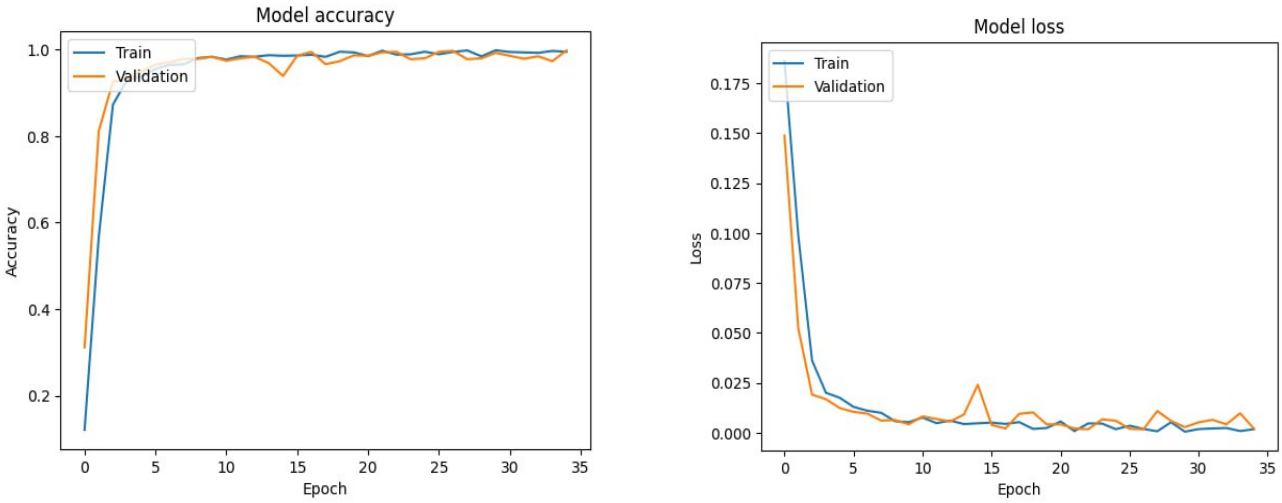


Figure 4. 8 Training and validation of RGB dataset 80-10-10 (A) Model accuracy (B) model loss

As we can see, the graph of the model's performance improvements over its 35-epoch training cycle reflects a robust learning process. Initially, the accuracy was low at 6.31%, but it quickly increased, achieving 85.04% by the third epoch and maintaining high performance throughout the subsequent epochs. The model's accuracy on the training set reached 99.24% by the final epoch, while the validation accuracy also showed significant gains, reaching 99.77%. The loss metrics indicated a continuous decrease, showcasing the model's effective minimization of errors: the training loss dropped from 0.2192 to 0.0024, and the validation loss decreased from 0.1488 to 0.0022. The final test accuracy was 99.66%. Despite occasional fluctuations in validation accuracy and loss, particularly between epochs 10 to 20, the overall trend demonstrated substantial convergence and model stability. These results suggest that the model was highly effective for the given task, without ove-0fitting and strong predictive performance.

Evaluating the model

The performance of the model is evaluated on the test data set in both RGB dataset 70-15-15 and RGB dataset 80-10-10 as follows.

Case 1: Here, the model trained on 70% of the dataset is evaluated. We have got 98.94% accuracy on the test dataset

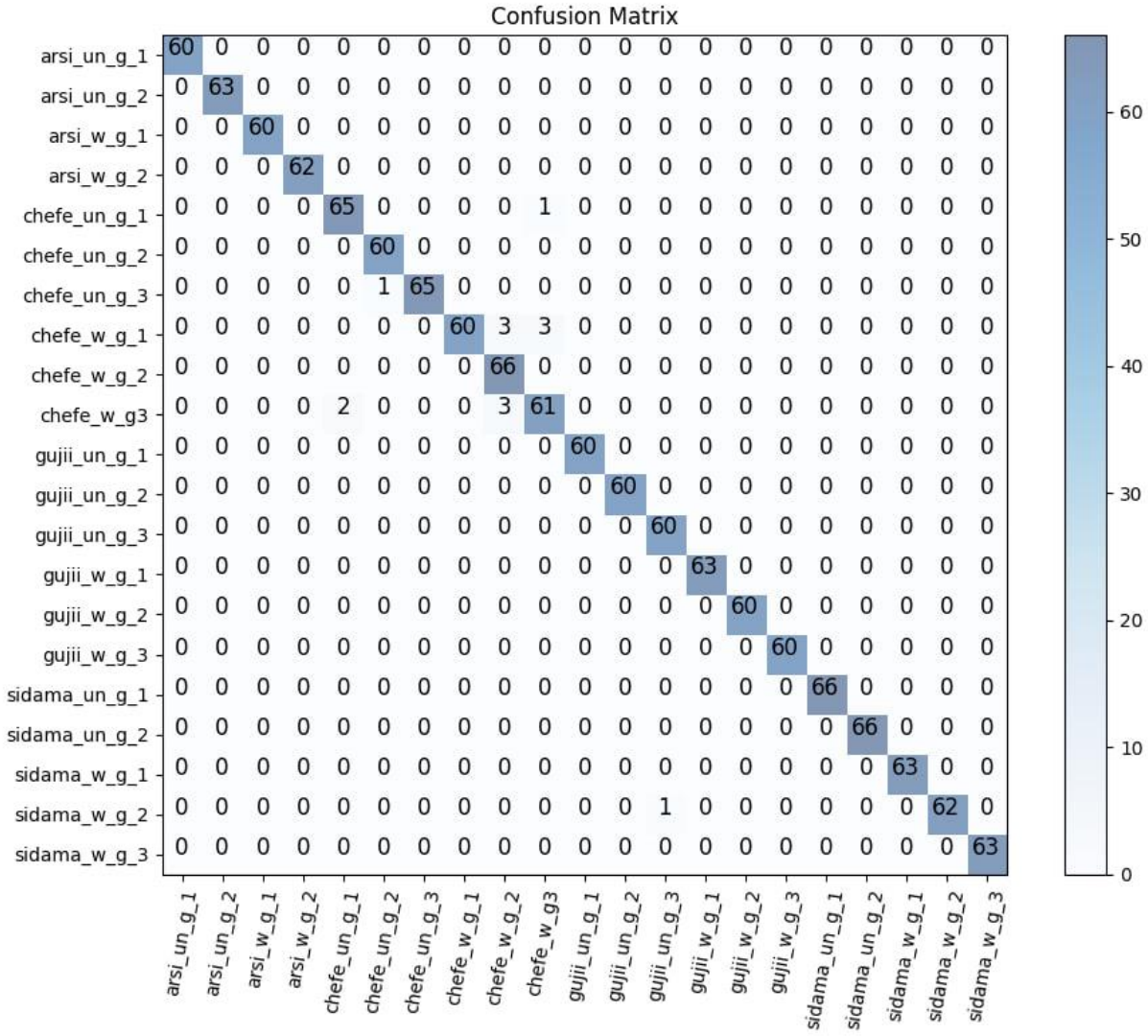


Figure4.9 Confusion matrix of RGB dataset 70-15-15

The confusion matrix and classification report reveal that the model excels in accurately predicting instances for the majority of classes, with true positive counts being high across the board. Most classes, such as the various Arsi, Gujii, and Sidama groups, exhibit perfect classification results with no misclassifications. However, some confusion exists in the Chefe classes, where a few instances are misclassified, resulting in minor false positive and false negative counts. For instance, chefe_w_g_1 has 3 false positives, and chefe_w_g_3 has both false positives and false negatives. Despite these few misclassifications, the overall performance of the model remains robust and highly reliable, demonstrating strong predictive accuracy and consistency.

Classification Report:

	precision	recall	f1-score	support
arsi_un_g_1	1.00	1.00	1.00	60
arsi_un_g_2	1.00	1.00	1.00	63
arsi_w_g_1	1.00	1.00	1.00	60
arsi_w_g_2	1.00	1.00	1.00	62
chefe_un_g_1	0.97	0.98	0.98	66
chefe_un_g_2	0.98	1.00	0.99	60
chefe_un_g_3	1.00	0.98	0.99	66
chefe_w_g_1	1.00	0.91	0.95	66
chefe_w_g_2	0.92	1.00	0.96	66
chefe_w_g3	0.94	0.92	0.93	66
gujii_un_g_1	1.00	1.00	1.00	60
gujii_un_g_2	1.00	1.00	1.00	60
gujii_un_g_3	0.98	1.00	0.99	60
gujii_w_g_1	1.00	1.00	1.00	63
gujii_w_g_2	1.00	1.00	1.00	60
gujii_w_g_3	1.00	1.00	1.00	60
sidama_un_g_1	1.00	1.00	1.00	66
sidama_un_g_2	1.00	1.00	1.00	66
sidama_w_g_1	1.00	1.00	1.00	63
sidama_w_g_2	1.00	0.98	0.99	63
sidama_w_g_3	1.00	1.00	1.00	63
accuracy			0.99	1319
macro avg	0.99	0.99	0.99	1319
weighted avg	0.99	0.99	0.99	1319

Figure 4.10 Classification Report of RGB dataset 70-15-15

The classification report demonstrates an overall excellent performance of the model with a near-perfect accuracy of 99%. Each of the 21 classes exhibited high precision, recall, and f1-scores, indicating that the model is highly effective in correctly identifying instances of each class. Specifically, classes such as arsi_un_g_1, arsi_un_g_2, arsi_w_g_1, arsi_w_g_2, and several others achieved perfect scores across all metrics, highlighting flawless classification in these categories. Although a few classes like chefe_w_g_1, chefe_w_g_2, and chefe_w_g3 had slightly lower f1-scores ranging from 0.93 to 0.96, they still maintained high performance with precision and recall close to or at 1.00. The macro and weighted averages for precision, recall, and f1-score were all 0.99, further affirming the model's consistent and reliable performance across the entire dataset of 1319 instances.

Case 2: Here, the model trained on 80% of the dataset is evaluated. We have got 99.66% accuracy on the test dataset

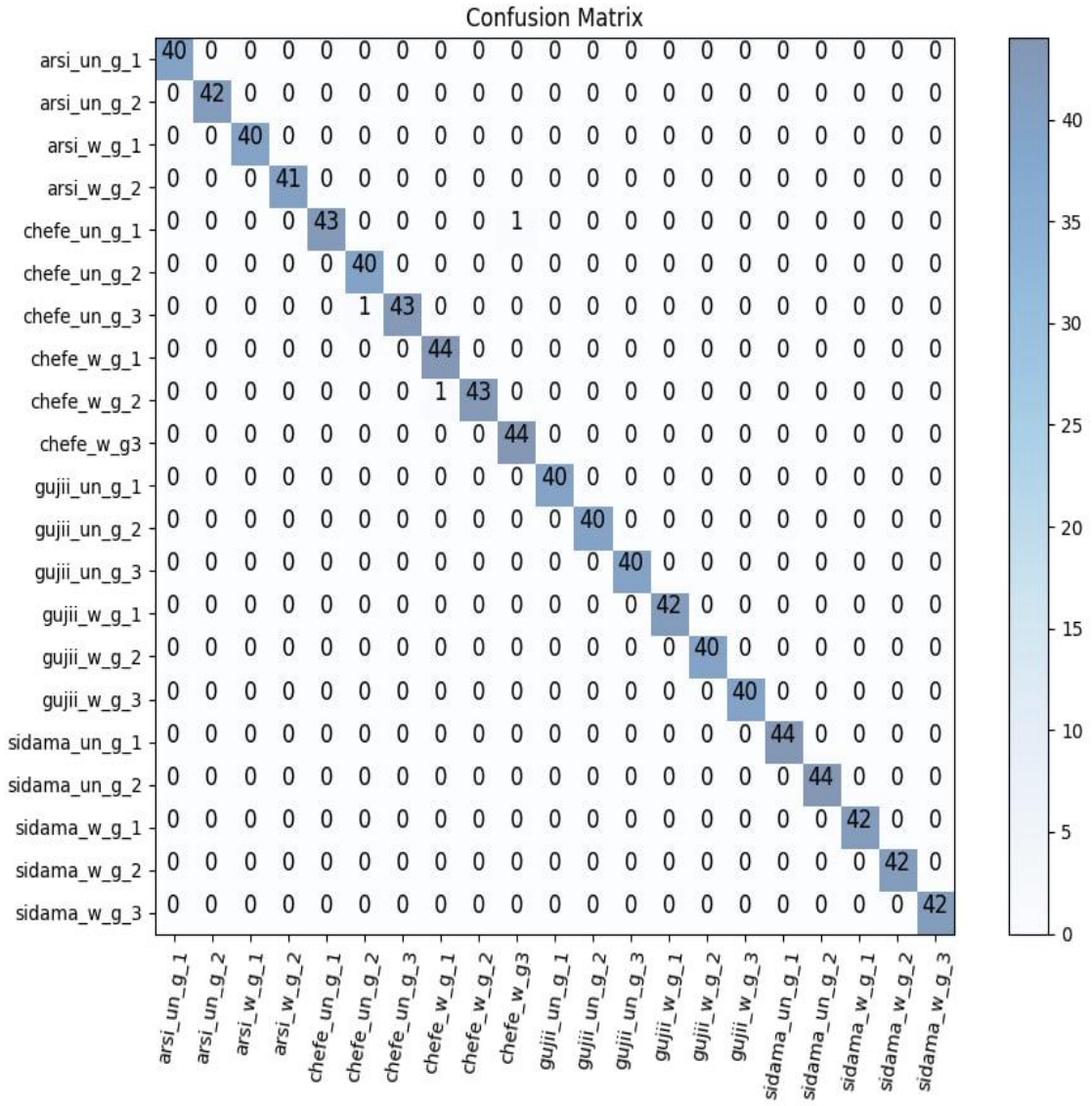


Figure 4. 11 Confusion matrix of RGB dataset 80-10-10

The confusion matrix represents the image illustrates the classification performance of a model across various classes, specifically within the categories of arsi, chefe, gujii, and sidama, each further subdivided into groups such as un_g_1, un_g_2, un_g_3, w_g_1, w_g_2, and w_g_3. Each cell in the matrix represents the number of predictions made by the model for a particular true class (rows) against the predicted class (columns). A clear pattern of strong diagonal dominance is evident, indicating high classification accuracy. For example, 'arsi_un_g_1' was correctly classified 40 times, 'chefe_un_g_2' 43 times, and 'sidama_w_g_3' 42 times, among others. These correct predictions along the diagonal show that the model performs well in correctly identifying samples from each specific group. However, there is minimal misclassification, with the off-

diagonal elements predominantly being zero, suggesting that there are very few instances where samples from one group were incorrectly predicted as belonging to another. The very few misclassifications (e.g., 'chefe_w_g_1' was predicted once as 'chefe_un_g_2') are isolated, indicating the robustness of the model in distinguishing between the different groups. Generally, this confusion matrix reflects a highly accurate model with a strong ability to correctly classify instances across multiple classes with minimal error. This performance underscores the effectiveness of the model in handling the given classification task, making it a reliable tool for the given dataset.

Classification Report:

	precision	recall	f1-score	support
arsi_un_g_1	1.00	1.00	1.00	40
arsi_un_g_2	1.00	1.00	1.00	42
arsi_w_g_1	1.00	1.00	1.00	40
arsi_w_g_2	1.00	1.00	1.00	41
chefe_un_g_1	1.00	0.98	0.99	44
chefe_un_g_2	0.98	1.00	0.99	40
chefe_un_g_3	1.00	0.98	0.99	44
chefe_w_g_1	0.98	1.00	0.99	44
chefe_w_g_2	1.00	0.98	0.99	44
chefe_w_g3	0.98	1.00	0.99	44
gujii_un_g_1	1.00	1.00	1.00	40
gujii_un_g_2	1.00	1.00	1.00	40
gujii_un_g_3	1.00	1.00	1.00	40
gujii_w_g_1	1.00	1.00	1.00	42
gujii_w_g_2	1.00	1.00	1.00	40
gujii_w_g_3	1.00	1.00	1.00	40
sidama_un_g_1	1.00	1.00	1.00	44
sidama_un_g_2	1.00	1.00	1.00	44
sidama_w_g_1	1.00	1.00	1.00	42
sidama_w_g_2	1.00	1.00	1.00	42
sidama_w_g_3	1.00	1.00	1.00	42
accuracy			1.00	879
macro avg	1.00	1.00	1.00	879
weighted avg	1.00	1.00	1.00	879

Figure 4. 12 Classification Report of RGB dataset 80-10-10

The classification report provides a comprehensive evaluation of the model's performance across 21 different classes, detailing metrics such as precision, recall, and F1-score, alongside the support for each class. Precision measures the accuracy of the positive predictions, and in this report, the precision scores are nearly perfect across all classes, predominantly at 1.00, with minor exceptions such as 'chefe_un_g_2' and 'chefe_w_g3,' which both have precision values of 0.98. This indicates that the model's predictions for these classes are highly reliable, with very few false positives.

Recall, which measures the ability of the model to capture all positive instances, also shows exemplary performance. Almost all classes have a recall of 1.00, with minor deviations observed in 'chefe_un_g_1,' 'chefe_un_g_3,' and 'chefe_w_g_2,' each having a recall of 0.98. This suggests that the model successfully identifies almost all actual positive cases, with very few false negatives. The F1-score, which is the harmonic mean of precision and recall, consolidates the model's overall classification performance. It remains at 1.00 for most classes, reflecting a balanced and effective classifier. a few instances where the F1-score is 0.99, such as in 'chefe_un_g_1,' 'chefe_un_g_3,' and 'chefe_w_g_2,' are indicative of the model's minor yet present imperfections in balancing precision and recall. The support values, which denote the number of actual occurrences for each class, vary slightly but are generally consistent, with each class having support between 40 to 44 instances. This balanced representation of classes in the dataset further strengthens the reliability of the performance metrics provided. The overall accuracy of the model stands at a stellar 1.00, highlighting the model's remarkable performance across all classes in the dataset. This is further affirmed by the macro average and weighted average values for precision, recall, and F1-score, all being 1.00. These averages consider both the individual class performance and the class distribution, underlining that the model performs uniformly well across all classes.

4.3. Discussion of Results

Dataset Configurations and Methods

We evaluated the performance of a Convolutional Neural Network (CNN) with Dropout layers, optimized using the Adam optimizer, on grayscale and RGB images of coffee beans. The datasets were split into training, validation, and testing sets using two configurations: 70-15-15 and 80-10-10. The results are summarized in the table4.6 below:

Table 4.2 Evaluation Summary

Dataset Configuration	**Method**	**Optimizer**	**Accuracy**
Grayscale coffee bean images (70-15-15)	CNN+ Dropout	Adam	98.56%
Grayscale coffee bean images (80-10-10)	CNN+ Dropout	Adam	99.09%
RGB coffee bean images (70-15-15)	CNN+ Dropout	Adam	98.94%
RGB coffee bean images (80-10-10)	CNN+ Dropout	Adam	99.66%
Total			99.06%

Research Question 1: Which image features, grayscale or RGB, are most effective in categorizing the quality and geographical origin of Ethiopian coffee beans using deep learning techniques?

Our findings clearly indicate that **RGB images** are more effective than grayscale images in categorizing the quality and geographical origin of Ethiopian coffee beans using deep learning techniques. The models trained on RGB images consistently outperformed those trained on grayscale images across all dataset configurations, with the highest accuracy of **99.66%** achieved using an 80-10-10 split of the RGB dataset.

This suggests that the additional color information present in RGB images plays a significant role in enhancing the model's ability to extract and learn relevant features, particularly those associated with coffee bean color, which is a critical determinant of quality and origin. The superior performance of RGB images underscores the importance of utilizing detailed image data in classification tasks, as it provides the model with richer feature sets that contribute to more accurate predictions.

Research Question 2: To what extent does the CNN model accurately identify the source and quality of Ethiopian coffee beans when applied to a multi-label classification task?

The deep learning algorithm demonstrated a high level of accuracy in identifying the source and quality of Ethiopian coffee beans when applied to a multi-label classification task. Across all

configurations and image types, the model achieved an overall average accuracy of **99.06%**, with the highest accuracy observed at **99.66%** for the RGB dataset with an 80-10-10 split.

These results indicate that the algorithm is highly effective at multi-label classification tasks, successfully identifying both the quality and geographical origin of the coffee beans. The consistent performance across different dataset splits also highlights the model's robustness and generalization capability, particularly when a larger proportion of the data is used for training (as seen in the 80-10-10 split).

Generally, the findings from our study strongly support the effectiveness of deep learning, particularly through the use of CNNs with Dropout layers and the Adam optimizer, in categorizing the quality and geographical origin of Ethiopian coffee beans. The results also underscore the importance of using RGB images and an enough amount of training data for optimal model performance. The study successfully answers both research questions, demonstrating the potential of deep learning models to automate and enhance the assessment and categorization processes in the coffee industry.

4.4 Environmental and Experimental Setting

To conduct the experiments on the coffee bean images, we have used two different environments. In the pre-processing stage (naming of coffee bean images, renaming, and resizing), we used a local machine with a core-i7, 16GB RAM, 1TB hard disk 8th generation laptop. Jupyter Notebook with Python 3.11 IDE was used for this purpose. The choice of the Jupyter Notebook was driven by its flexibility and control over the local computing environment. Despite the local hardware limitations, we found Jupyter Notebook to be faster than Google Colab during model training in our specific case. This performance can be attributed to the optimized configuration and absence of network latency, which can affect cloud-based environments like Google Colab.

CHAPTER FIVE

5 Conclusion and Recommendation

5.1 Conclusion

In conclusion, this study demonstrates the effectiveness of deep learning, particularly Convolutional Neural Networks (CNNs), in automating the classification of coffee beans by quality and geographical origin. By collecting an extensive dataset of 3,965 coffee beans from the Ethiopian regions of Arsi, Yirgacheffe, Guji, and Sidama, and capturing 6,373 images (front and back of each bean), we created a comprehensive dataset that was further balanced to include 8,790 images. Additionally, larger training datasets contribute to better model generalization, as shown by the superior performance with an 80-10-10 training split. Our experiments revealed that models trained on RGB images achieved superior performance, with a peak accuracy of 99.66% and an average accuracy of 99.06% across all experiments, highlighting the significance of color information in image-based classification tasks. The use of Dropout layers and the Adam optimizer proved crucial in preventing over-fitting and ensuring efficient model training. These findings validate the potential of deploying CNNs in real-world coffee industry applications, offering a scalable and accurate solution for the classification of coffee beans. This technological advancement promises to enhance quality control processes, improve consistency, and ultimately increase consumer satisfaction and market value for coffee producers.

5.2 Recommendations

Based on the findings of this study, I recommend the following for future research and practical applications in the coffee industry:

- Future research should focus on expanding the dataset to include a wider variety of coffee beans from different regions and of different qualities. This will help create more robust models that can be generalized across a broader spectrum of coffee types.
- Investigating advanced deep learning techniques such as transfer learning and ensemble methods could further improve classification accuracy and model robustness. These techniques can leverage pre-trained models and combine multiple models to enhance performance.

- Efforts should be made to implement these models in real-time coffee quality assessment systems. This could involve developing mobile applications or integrating the models into existing coffee processing machinery to provide instant quality feedback.
- The techniques developed in this study could be adapted for use in other agricultural sectors where quality assessment of products is crucial. Applying these models to different crops could provide similar benefits in terms of automation and accuracy.
- To maintain high accuracy and relevance, it is recommended to continuously update the models with new data. This includes incorporating images of coffee beans from new harvests and different growing conditions to ensure the models remain current and effective.

6. Reference

[1] M.-C. Combes, T. Joët, And P. Lashermes, “Development Of A Rapid And Efficient Dna-Based Method To Detect And Quantify Adulterations In Coffee (Arabica Versus Robusta),” *Food Control*, Vol. 88, Pp. 198–206, Jun. 2018, Doi: 10.1016/J.Foodcont.2018.01.014.

[2] K. M. Ting, “Confusion Matrix,” In *Encyclopedia Of Machine Learning And Data Mining*, C. Sammut And G. I. Webb, Eds., Boston, Ma: Springer Us, 2017, Pp. 260–260. Doi: 10.1007/978-1-4899-7687-1_50.

[3] K. M. Ting, “Confusion Matrix,” In *Encyclopedia Of Machine Learning*, C. Sammut And G. I. Webb, Eds., Boston, Ma: Springer Us, 2010, Pp. 209–209. Doi: 10.1007/978-0-387-30164-8_157.

[4] “Food Security And Nutrition And Sustainable Agriculture | Department Of Economic And Social Affairs.” Accessed: Aug. 20, 2024. [Online]. Available: Https://Sdgs.Un.Org/Topics/Food-Security-And-Nutrition-And-Sustainable-Agriculture

[5] “Agriculture And Food,” World Bank. Accessed: Aug. 20, 2024. [Online]. Available: Https://Www.Worldbank.Org/En/Topic/Agriculture/Overview

[6] “Agriculture Benefits And Its Role.Pdf.”

[7] “Coffee Consumption By Country 2023,” Wisevoter. Accessed: Oct. 18, 2023. [Online]. Available: Https://Wisevoter.Com/Country-Rankings/Coffee-Consumption-By-Country/

[8] “Coffee Market In Ethiopia - Industry Report & Analysis.” Accessed: Jan. 11, 2024. [Online]. Available: Https://Www.Mordorintelligence.Com/Industry-Reports/Ethiopia-Coffee-Market

[9] J. N. Barbosa, F. M. Borem, M. A. Cirillo, M. R. Malta, A. A. Alvarenga, And H. M. R. Alves, “Coffee Quality And Its Interactions With Environmental Factors In Minas Gerais, Brazil,” *J. Agric. Sci.*, Vol. 4, No. 5, P. P181, Mar. 2012, Doi: 10.5539/Jas.V4n5p181.

[10] “The Origin & Story Of Ethiopian Coffee,” Https://Cornerperk.Com/. Accessed: Mar. 19, 2024. [Online]. Available: Https://Cornerperk.Com/The-Origin-Story-Of-Ethiopian-Coffee/

[11] T. Debela, “Classification Of Genetically Variable Wollega Coffee Beans Using Imaging Techniques And Artificial Neural Network,” 2023.

[12] G. Amtate And D. Teferi, “Multiclass Classification Of Ethiopian Coffee Bean Using Deep Learning,” *Sinet Ethiop. J. Sci.*, Vol. 45, No. 3, Pp. 309–321, Dec. 2022, Doi: 10.4314/Sinet.V45i3.6.

[13] “Ecta-New-Coffee-Booklet-.Pdf.” Accessed: Jan. 17, 2024. [Online]. Available: Https://Ethiocta.Gov.Et/Wp-Content/Uploads/2022/09/Ecta-New-Coffee-Booklet-.Pdf

[14] “Ethiopia Coffee Reviews And Ratings,” Coffee Review. Accessed: Jan. 23, 2024. [Online]. Available: Https://Www.Coffeereview.Com/Coffee-Origins/Ethiopia/

[15] Denahaines, “Ethiopian Coffee Guide: Beans, Regions, Flavor Notes, Origins | Enjoyjava.” Accessed: Jan. 12, 2024. [Online]. Available: Https://Enjoyjava.Com/Ethiopian-Coffee/

[16] “5 Key Differences Between Washed & Natural Process Ethiopian Coffee – Mikro Coffee Roasters Torquay.” Accessed: Jan. 23, 2024. [Online]. Available: Https://Mikro.Coffee/Blogs/Australian-Coffee-News/5-Key-Differences-Between-Washed-Natural-Process-Ethiopian-Coffee

[17] H. M. Aycheh, “Image Analysis For Ethiopian Coffee Classification”.

[18] “Image Processing Based Disease Detection For Sugarcane Leaves.” Accessed: Mar. 19, 2024. [Online]. Available: Https://Www.Jetir.Org/View?Paper=Jetir1806177

[19] C. Luo, Y. Hao, And Z. Tong, “Research On Digital Image Processing Technology And Its Application,” *Adv. Intell. Syst. Res.*, Vol. 163.

[20] Dbit, “Research : Department Of Information Science And Engineering,” >. Accessed: Mar. 19, 2024. [Online]. Available: Https://Dbit.Co.In/Programs/Under-Graduate/Information-Science/Research

[21] “(12) Image Processing | Linkedin.” Accessed: Mar. 19, 2024. [Online]. Available: Https://Www.Linkedin.Com/Pulse/Image-Processing-Dipti-Goyal-1f/

[22] Vaibhavraheja, “Computer Vision In Lane And Object Detection,” Medium. Accessed: Mar. 19, 2024. [Online]. Available: Https://Medium.Com/@Vaibhavraheja32/Computer-Vision-In-Lane-And-Object-Detection-40ac31aab667

[23] "(12) Here's All That You Need To Know About Image Processing | Linkedin." Accessed: Mar. 19, 2024. [Online]. Available: Https://Www.Linkedin.Com/Pulse/Heres-All-You-Need-Know-Image-Processing-Bytrix-Technologies/

[24] G. Anbarjafari, *1. Introduction To Image Processing*. Accessed: Feb. 07, 2024. [Online]. Available: Https://Sisu.Ut.Ee/Imageprocessing/Book/1

[25] G. Anbarjafari, *1. Introduction To Image Processing*. Accessed: Mar. 19, 2024. [Online]. Available: Https://Sisu.Ut.Ee/Imageprocessing/Book/1

[26] "Segmentation.Docx - Image Segmentation Is A Fundamental Task In Computer Vision That - College Sidekick." Accessed: Mar. 19, 2024. [Online]. Available: Https://Www.Collegesidekick.Com/Study-Docs/5439221

[27] A. B. Emiru, "Bahir Dar University Bahir Dar Institute Of Technology School Of Graduate Studies Faculty Of Computing Msc. Thesis On Automatic Identification Of Faba Bean Varieties Using Machine Learning Approach".

[28] "3. Artificial Neural Networks - Machine Learning And Data Science Blueprints For Finance [Book]." Accessed: Mar. 19, 2024. [Online]. Available: Https://Www.Oreilly.Com/Library/View/Machine-Learning-And/9781492073048/Ch03.Html

[29] "A Neural Network? - Caltech." Accessed: Aug. 21, 2024. [Online]. Available: Https://Pg-P.Ctme.Caltech.Edu/Blog/Ai-Ml/What-Is-A-Neural-Network

[30] M. L. In P. English, "Neural Network," Medium. Accessed: Mar. 22, 2024. [Online]. Available: Https://Medium.Com/@Nerdjock/Convolutional-Neural-Network-Lesson-5-Padding-98b50660ddbd

[31] M. Saeed, "An Introduction To Recurrent Neural Networks And The Math That Powers Them," Machinelearningmastery.Com. Accessed: Aug. 21, 2024. [Online]. Available: Https://Machinelearningmastery.Com/An-Introduction-To-Recurrent-Neural-Networks-And-The-Math-That-Powers-Them/

[32] K. Raval, "Q/A System — Deep Learning(1/2)," Medium. Accessed: Mar. 19, 2024. [Online]. Available: Https://Medium.Com/@Kashyapraval/Qna-System-Deep-Learning-1-2-4aa20c017042

[33] R. Yamashita, M. Nishio, R. K. G. Do, And K. Togashi, “Convolutional Neural Networks: An Overview And Application In Radiology,” *Insights Imaging*, Vol. 9, No. 4, Art. No. 4, Aug. 2018, Doi: 10.1007/S13244-018-0639-9.

[34] K. G. Kim, “Book Review: Deep Learning,” *Healthc. Inform. Res.*, Vol. 22, No. 4, Pp. 351–354, Oct. 2016, Doi: 10.4258/Hir.2016.22.4.351.

[35] G. Adeneye, “Research Guides: Interactive Media: Home.” Accessed: Mar. 19, 2024. [Online]. Available: Https://Guides.Nyu.Edu/C.Php?G=814764&P=5814404

[36] A. Voulodimos, N. Doulamis, A. Doulamis, And E. Protopapadakis, “Deep Learning For Computer Vision: A Brief Review,” *Comput. Intell. Neurosci.*, Vol. 2018, P. E7068349, Feb. 2018, Doi: 10.1155/2018/7068349.

[37] A. Voulodimos, N. Doulamis, A. Doulamis, And E. Protopapadakis, “Deep Learning For Computer Vision: A Brief Review,” *Comput. Intell. Neurosci.*, Vol. 2018, P. 7068349, Feb. 2018, Doi: 10.1155/2018/7068349.

[38] R. Yamashita, M. Nishio, R. K. G. Do, And K. Togashi, “Convolutional Neural Networks: An Overview And Application In Radiology,” *Insights Imaging*, Vol. 9, No. 4, Pp. 611–629, Jun. 2018, Doi: 10.1007/S13244-018-0639-9.

[39] R. Chauhan, K. K. Ghanshala, And R. C. Joshi, “Convolutional Neural Network (Cnn) For Image Detection And Recognition,” In *2018 First International Conference On Secure Cyber Computing And Communication (Icsccc)*, Jalandhar, India: Ieee, Dec. 2018, Pp. 278–282. Doi: 10.1109/Icsccc.2018.8703316.

[40] S. Sadashiva, “Testing The Effects Of Covid_19 On Lungs,” Medium. Accessed: Mar. 19, 2024. [Online]. Available: Https://Sowjanyasadashiva.Medium.Com/C-2caae0eab9b4

[41] “Convolutional Neural Networks 2.Pdf.”

[42] D. Buonocore, M. Carratù, And M. Lamberti, “Classification Of Coffee Bean Varieties Based On A Deep Learning Approach,” In *Proceedings Of The 18th Imeko Tc10 Conference On Measurement For Diagnostics, Optimisation And Control*, Warsaw, Poland: Imeko, 2023, Pp. 14–19. Doi: 10.21014/Tc10-2022.002.

[43] C. Pinto, J. Furukawa, H. Fukai, And S. Tamura, “Classification Of Green Coffee Bean Images Basec On Defect Types Using Convolutional Neural Network (Cnn),” In *2017 International Conference On Advanced Informatics, Concepts, Theory, And Applications (Icaicta)*, Denpasar: Ieee, Aug. 2017, Pp. 1–5. Doi: 10.1109/Icaicta.2017.8090980.

[44] J. N. C. Sarino, M. M. Bayas, E. R. Arboleda, E. C. Guevarra, And R. M. Dellosa, “Classification Of Coffee Bean Degree Of Roast Using Image Processing And Neural Network,” Vol. 8, No. 10, 2019.

[45] J. Y. Kim, “Coffee Beans Quality Prediction Using Machine Learning”.

[46] M. S. Fuentes, N. A. L. Zelaya, And J. L. O. Avila, “Coffee Fruit Recognition Using Artificial Vision And Neural Networks,” In *2020 5th International Conference On Control And Robotics Engineering (Iccre)*, Osaka, Japan: Ieee, Apr. 2020, Pp. 224–228. Doi: 10.1109/Iccre49379.2020.9096441.

[47] S. Wallelign, M. Polceanu, T. Jemal, And C. Buche, “Coffee Grading With Convolutional Neural Networks Using Small Datasets With High Variance,” *J. Wscg*, Vol. 27, No. 2, 2019, Doi: 10.24132/Jwscg.2019.27.2.4.

Appendix : Sample code

1. Python code to the file from the specific folder

```
# Load the CSV files
train_df = pd.read_csv('D:\\data\\splited_data\\Data70_15_15\\train_dataset.csv')
validation_df = pd.read_csv('D:\\data\\splited_data\\Data70_15_15\\validation_dataset.csv')
test_df = pd.read_csv('D:\\data\\splited_data\\Data70_15_15\\test_dataset.csv')


# Function to load images and resize them
def load_images(df, target_size=(224, 224)):
    images = []
    labels = []
    for _, row in df.iterrows():
        image = tf.keras.preprocessing.image.load_img(row['image_path']
,color_mode='grayscale',target_size=target_size)
        image = tf.keras.preprocessing.image.img_to_array(image)
        images.append(image)
        labels.append(row[['arsi_un_g_1', 'arsi_un_g_2', 'arsi_w_g_1', 'arsi_w_g_2',
                           'chefe_un_g_1', 'chefe_un_g_2', 'chefe_un_g_3', 'chefe_w_g_1',
                           'chefe_w_g_2', 'chefe_w_g3', 'gujii_un_g_1', 'gujii_un_g_2',
                           'gujii_un_g_3', 'gujii_w_g_1', 'gujii_w_g_2', 'gujii_w_g_3',
                           'sidama_un_g_1', 'sidama_un_g_2', 'sidama_w_g_1', 'sidama_w_g_2',
                           'sidama_w_g_3']].values.astype(np.float32))
    return np.array(images), np.array(labels)
```

2. Python code to design the CNN model using TensorFlow high-level Keras API

```
# Define the model architecture
model = Sequential([
   Conv2D(32, (3, 3), activation='relu',padding='same' ,input_shape=(224, 224, 3)),
   MaxPooling2D((2, 2)),
   Conv2D(64, (3, 3), activation='relu' ,padding='same'),
   MaxPooling2D((2, 2)),
   Conv2D(128, (3, 3), activation='relu',padding='same'),
   MaxPooling2D((2, 2)),
   Conv2D(128, (3, 3), activation='relu',padding='same'),
   MaxPooling2D((2, 2)),
   Flatten(),
   Dense(512, activation='relu'),
   Dropout(0.4),
   Dense(256, activation='relu'),
   Dropout(0.2),
   Dense(len(train_labels[0]), activation='sigmoid')
])
```

3. Python Code to Compile the model

```
# Compile the model
model.compile(optimizer='adam',
              loss='binary_crossentropy',
              metrics=['accuracy'])
model.summary()
```

4. Python code to train the model

```
# Train the model
history=model.fit(train_images, train_labels, epochs=35,
        validation_data=(validation_images, validation_labels))


# Evaluate the model on the test set
test_loss, test_acc = model.evaluate(test_images, test_labels)
print('Test accuracy:', test_acc)
```

5. Python code to evaluate the model performance on the test dataset

```
# Calculate the classification report and confusion matrix
from sklearn.metrics import classification_report, confusion_matrix
class_labels = ['arsi_un_g_1', 'arsi_un_g_2', 'arsi_w_g_1', 'arsi_w_g_2',
                'chefe_un_g_1', 'chefe_un_g_2', 'chefe_un_g_3', 'chefe_w_g_1',
                'chefe_w_g_2', 'chefe_w_g3', 'gujii_un_g_1', 'gujii_un_g_2',
                'gujii_un_g_3', 'gujii_w_g_1', 'gujii_w_g_2', 'gujii_w_g_3',
                'sidama_un_g_1', 'sidama_un_g_2', 'sidama_w_g_1', 'sidama_w_g_2',
                'sidama_w_g_3']
report = classification_report(np.argmax(test_labels, axis=1), np.argmax(predictions,
axis=1), target_names=class_labels)
print('Classification Report:')
print(report)
conf_mat = confusion_matrix(np.argmax(test_labels, axis=1), np.argmax(predictions,
axis=1))


# Plot the confusion matrix using matplotlib
plt.figure(figsize=(10,8))
plt.imshow(conf_mat, interpolation='nearest', cmap='Blues', alpha=0.5)
plt.title("Confusion Matrix")
plt.colorbar()
tick_marks = np.arange(len(class_labels))
plt.xticks(tick_marks, class_labels,rotation=80)
```

```
plt.yticks(tick_marks, class_labels)
for i in range(conf_mat.shape[0]):
    for j in range(conf_mat.shape[1]):
        plt.text(j, i, str(conf_mat[i, j]), fontsize=12, ha="center")
plt.tight_layout()
plt.show()
```