# DILLA UNIVERSITY

## COLLEGE OF ENGINEERING &TECHNOLOGY

## SCHOOL OF COMPUTING & INFORMATICS

M.Sc in Computer Science & Networking

**Thesis** : Detection and Classification of Wheat Rust Disease Using Deep Learning


**By**

**ANCHINESH MOLLA**

**Advisor:** DEGIF TEKA (PhD)


Thesis Submitted to the School of Computing and Informatics in Meeting the Preliminary Thesis Requirement for Partial Fulfillment of the Award of Master of Degree in Computer Science and Networking.

**Dilla,Ethiopia**

**2023**


# DILLA UNIVERSITY

## COLLEGE OF ENGINEERING &TECHNOLOGY

## SCHOOL OF COMPUTING &INFORMATICS

Declaration

I, the undersigned, declare that this thesis is my original work and has not been presented for a degree at any other university, and that all sources of materials used for the thesis have been dulyacknowledged.

Declared by:

Name: _____

Signature: _____

Date: _____

Confirmed by the advisor:

Name: _____

Signature: _____

Date: _____


# DILLA UNIVERSITY

## COLLEGE OF ENGINEERING &TECHNOLOGY

## SCHOOL OF COMPUTING &INFORMATICS

This is to certify that the thesis prepared by Anchinesh Molla, entitled "Detection and Classification of Wheat Rust Disease Using Deep Learning," submitted in partial fulfillment of the requirement for the degree of Master of Science in Computer Science and Networking, complies with the regulations of the university and meets the accepted standards concerning originality and quality.

**Signature of the Board of Examiners for Approval**

| | Name | Signature | Date |
|---|---|---|---|
| Advisor: | _____ | _____ | _____ |
| Chairman: | _____ | _____ | _____ |
| Internal Examiner: | _____ | _____ | _____ |
| External Examiner: | _____ | _____ | _____ |

## Acknowledgment

First and foremost, I would like to thank my creator and savior, God, for giving me the moral, psychological, and spiritual strength to accomplish this thesis work.

Secondly, I would like to extend my gratitude to my advisor, Dr. Degif Teka, for his vital effort in advising, guidance, suggestions, ideas, checking, and providing knowledgeable, constructive, and valuable comments starting from the time of title selection up until the time of completion of this work.

Thirdly, I would like to express my sincere gratitude to my mom for all the indescribable sacrifices she made so that I have become who I am today.

I would like to extend my special appreciation to my brother and assistant professor, Alamayehu Molla, who covered my tuition fees, paid for my cost sharing, and provided endless support that never stopped helping me financially and mentally.

I also would like to express my special gratitude to Mr. Kelelaw, lecturer and plant science researcher at Injjebera University; Mr. Getenet, plant science expert at Arebaminche Agricultural Centre; and Adet Agricultural Research Centre experts for their help in obtaining data and for providing me with information about wheat rust diseases. With their passionate participation and input, the data collection has been successfully conducted.

Finally, I am deeply grateful to my brother and Dr. Yenew Molla, my family, and my best friends Tilahun and Denkie for their endless support and encouragement through this thesis.

## Abstract

The wheat crop is among the most significant staple cereal crops in the world. Ethiopia is first in both area and area output of wheat in sub-Saharan Africa, with a share of 55.5% and 47.82% capacity to become a regional exporter. The most prevalent diseases in wheat are wheat leaf rust, wheat yellow rust, and wheat stem rust. It affects the production and quality of wheat all over the world. Because rust disease spreads rapidly in a matter of days, early detection is a difficult task. The research problem addressed in this study is the lack of accurate and efficient methods for early detection and classification of wheat rust disease. Traditional methods of detecting and classifying wheat rust diseases are time-consuming and less accurate than deep learning techniques. Lack of studies that explore the use of deep learning techniques for the detection and classification of wheat rust disease in Ethiopia are research gaps we identified. The study aimed to fill this gap by exploring the effectiveness of deep learning techniques for this purpose. Adet and Arebaminch agricultural research centers were the institutes in Ethiopia where the images of wheat leaf and stem were taken. A total of 2538 images were gathered and compiled for this study we have 15,318 total dataset after augmentation. The suggested system incorporates image preprocessing elements like resizing and augmentation. To accomplish the key goal in this study, pre-trained VGG-19, ResNet-50, and convolution neural network techniques are employed. Regularization methods like dropout, L1 regularization, and early stopping were proposed in this study to improve the model's effectiveness and accuracy. The experimental result shows that the test accuracy obtained from the transfer learning models VGG19 and ResNnet_50 achieves an accuracy of 81.7% and 84.8%, respectively by using our dataset. Furthermore, the proposed model outperforms in terms of performance and successfully detects and classifies the examined images with a training accuracy of 97.07.8% and a testing accuracy of 96.23% with the softmax activation function after applying the regularization method. The research contribution of this study is the development of accurate model for detecting and classifying wheat rust disease.

***Keywords:*** ***wheat rust disease, fine tuning, softmax, Convocational neural networks, transfer learning, deep learning***

Contents

# List of figures

# List of tables

## List of acronyms

| | |
|---|---|
| SNNP- | South Nations, Nationalities and Peoples |
| CNN– | Convolution neural networks |
| KNN – | K-nearest neighbor algorithm |
| DIP- | Digital Image Processing |
| ReLU- | Rectified linear units |
| SVM- | support vector machine |
| GPUs - | Graphical processing units |
| RNN- | Recurrent neural network |
| DBN – | Deep belief networks |
| ResNet – | Residual networks |
| MCC - | Matthews correlation coefficient |
| DCNN- | Deep Convolution neural networks |
| LSTM- | long short term memory networks |
| FPR- | false positive rate |
| LVQ - | Learning Vector Quantization algorithm |
| VGG- | Visual Geometry Group |
| RGB- | Red Green Blue |
| API - | application programming interface |

# CHAPTER ONE

## 1. INTRODUCTION

### 1.1 Background of the study

One of the most important crops and sources of food for people on every continent is wheat. The majority of global countries' populations depend on the production of wheat, which takes up the largest portion of planted land. Fungus infections are one of the key parameters influencing the yield of wheat[1]. Wheat growing seasons differ depending on where you are in the world. The major wheat season in most parts of the nation follows the lengthy rainy season (meher), which begins in June and ends in December. Wheat is also farmed in some locations during the previous short rainy (belg) season, which runs from March or April through June or July[2]. Despite wheat's economic potential for meal security, the actual output under small-scale farmer settings is poor due to a number of production challenges [3].

Some of the most common challenges to production include climate change, pollinator decline, and plant disease. Plant diseases are an issue for international food security; they also have disastrous repercussions for Ethiopian smallholder families that are responsible for maintaining many people in one household. The wheat crop is Ethiopia's third [4]] most important source of food security, accounting for 17 percent of the country's overall agricultural land utilization [3]. It is not only a vital crop for small-scale farmer's livelihoods but also provides safety for millions of Ethiopians.

In Ethiopia, the outbreak of the disease is endangering the country's crop production gains in wheat-producing regions. The environment in Ethiopia's primary wheat-producing areas is conducive to wheat rust infection virtually all year [2].

The most significant biotic barriers to Ethiopian wheat production are rust (fungal) diseases, particularly stem (black) and stripe (yellow) rust. In recent years, widespread output losses have been brought on by recurrent rust epidemics. Ethiopian wheat output is seriously and persistently threatened by the invasion and spread of new, aggressive wheat rust races. Rust resistance has therefore been given top priority in wheat modification programmers in Ethiopia as a crucial quality in the creation of new, improved varieties. Ethiopia views the breeding and propagation of new, improved varieties as a crucial aspect of agricultural growth[5]. Rust infections are fungi-

caused bacterial infections that affect plants, primarily grasses. In particular, they damage the leaf surfaces of aerial plants, although they can also harm stems, flowers, and even fruits. They have intricate life cycles that call for two distinct, unrelated hosts. The color of the spore pustules that rust creates depends on the kind of rust.

Due to the brown color of the round urediniospores on the surfaces of the crop's leaves, leaf rust is also known as brown rust. Yellow stripes on the surfaces of the leaf are a defining feature of yellow rust, often known as stripe rust. Additionally, brown stem rust is identified by brown spots on the outermost layer of the branches[6]. Wheat rust disease was first discovered in the Southern Nations (SNNP) and Oromia regions [7].

Recently, widespread output losses have been brought on by recurrent rust outbreaks. Ethiopian wheat output is seriously and persistently threatened by the invasion and spread of new, aggressive wheat rust species.

Smallholder farmers' livelihoods are in danger and they suffer economic losses as a result of fungus illnesses. Even though it is known that wheat rust epidemics have happened in Ethiopia, there hasn't been a thorough review of previous outbreaks in a long time. The fungus that causes wheat crop damage has since spread to numerous regional states of Ethiopia, particularly in Ethiopia's north, particularly in the Amhara and Tigray areas. These diseases are caused by funguses, which are widely recognized for stunting plants and resulting in pre-harvest losses of up to 100% in the worst cases[7]. On October 24, 2016, regional authorities discovered the disease in approximately 2200 localities across the country, covering nearly 300,000 hectares of land [8].

Ethiopia, one of Africa's top wheat producers, faces the greatest biological challenge to wheat output from wheat rust. Fungal diseases result in financial losses and threaten the livelihoods of smallholder farmers. Although Ethiopia has had wheat rust epidemics, no systematic long-term examination of earlier outbreaks has been conducted [2].

Since October 24, 2016, Ethiopian regional authorities have sprayed fungicides on an estimated 92 percent of wheat farms in the country that had been determined to be wheat rust infected, as a measure of the government's aggressive reaction [7]. However, if it is not inspected and action is not taken in a timely manner, it may continue to survive the fungicide.

Farmers and professionals who spray the fungicide face health and safety issues. Furthermore, many farms required multiple sprayings.

The advancement of machine learning techniques based on the usage of neural network algorithms has recently given the technologies for plant disease monitoring based on digital RGB images a substantial boost. Deep learning neural networks differ from conventional approaches in that their multilayer architecture of neurons allows the subsequent layer to draw conclusions about the objects under study using the output of the preceding layer as input information. Convolutional neural networks (CNNs), for instance, are among the most effective techniques for such a crucial task as picture labeling. CNNs can be built in a variety of ways[9].

This proposed model emphasizes the significance of creating an effective model in order to help these farmers detect and distinguish rusty leaves, which can then be used to benefit them. As a result, we developed a deep learning method to determine if a wheat crop is infected with one of the three types of wheat rust disease or is otherwise uninfected. This work presents a deep learning model for early detection and measurement in large government agricultural fields.

## 1.2 Motivation

Plant disease detection is a significant topic that has undergone extensive investigation and is driven by the desire to provide nutrient-rich food. Plants offer us food, shelter, fibre, medicine, and fuel. Modern technology, on the other hand, enables people to generate more food in order to meet their demands. Food safety is influenced by a variety of challenges, including climate change and disease-induced plant loss. Currently, in the agricultural industry, automatic detection of crop diseases is widely used around the world. This technology is quite restricted in our country. The most significant aspect of plant disease management is detecting the disease as soon as it emerges on the farm, which may be accomplished utilizing various computing infrastructures such as deep learning, machine learning, and computer vision techniques. As a result, we seen detection as a part of the solution to increasing agricultural productivity.

The key motive for pursuing this thesis was to contribute to the field by developing a model that can detect wheat rust disease quickly and accurately, allowing for prompt assessments before the disease causes entire wheat destruction. Furthermore, computer vision is now providing several benefits in a variety of industries, including medical diagnostics and remote sensing. The thesis proposed applying technology to identify wheat disease in order to boost wheat crop yields in

Ethiopia. CNN was applied to extract features automatically and enhance the mechanism of wheat rust detection. The main motive of this thesis was to develop a model that boosts wheat production by delivering automated information regarding infected and non-infected wheat leaves at an early stage. This thesis allows us to greatly improve the detection method in comparison to previous detection mechanisms.

## 1.3Statement of Problem

After teff and maize, wheat is the third most significant cereal crop in Ethiopia, contributing around 17% to total grain production [3]. However, the country's most challenging problem right now is the spread of numerous diseases on different crops, which has resulted in various socioeconomic problems, including food hunger, market price hikes, and a lack of money as a result of the need for imports from other countries to cover them. Wheat rust is the most significant biological factor in wheat production in Ethiopia, one of Africa's greatest wheat producers[3].

Additionally, complex and intricate interactions between biotic and abiotic elements, particularly in smallholder farming systems, have an impact on wheat yield. A number of major biotic variables, including stem rust, yellow rust, and leaf rust, have a significant impact on Ethiopia's wheat productivity [3]. Wheat rust is the most destructive and widely spreading disease in many parts of our country, notably in wheat-growing areas. For example, as we interviewed agricultural research experts at the agricultural research institute centers of Arbaminch and Adet, they said that wheat rust is a common problem in their institute. Wheat rust infections are a problem because they can significantly lower the quality and supply of wheat products. Wheat rust disease may now be detected with the naked eye by professionals. According to an interview we conducted with agriculture specialists and researchers, Ethiopia's largest difficulty is that most farmers are illiterate and are unable to detect disease at an early stage, necessitating expert advice. Without the use of technology, experts cannot reach across all disciplines at an early stage of the disease. To boost wheat production and productivity, farmers consult specialists for guidance on what to do if the disease strikes wheat.

naked eye based detection and classification is unsuccessful, labor-intensive, and time-consuming because crop pathologists are unable to visit every farm and must instead rely on manual eye assessment. Farmers might have to travel a significant distance to meet with professionals. Despite the fact that they travel such enormous distances, the professionals the farmer wishes to contact

may not be able to provide advice. Obtaining expert guidance in this matter is both costly and time-consuming. In addition, whenever professionals are doubtful whether a disease has occurred, they collect a sample of the disease and test it in a laboratory to determine its identity. The existing disease identification process has the drawback of iterating until the type of disease evaluated in the laboratory is certain. Consequently, due to the disease's significant impact on wheat production, the method is normally slow, error-prone, and ineffective. As a result, the country's wheat production efficiency and effectiveness decreased. Continuous monitoring and early diagnosis of disease on a large farm are challenging, resulting in a major outbreak that cannot be controlled organically.

Most research in the area of image processing uses handcrafted feature extraction techniques to detect yellow rust in wheat, followed by traditional machine learning classifiers, including fuzzy algorithms, support vector machines, and the KNN algorithm [10]. There are a number of distinctions between these techniques and deep learning methods. Deep learning does not require handcrafted feature extraction. But traditional methods for classifying images relied on manually created features, whose effectiveness and accuracy had a significant impact on the final outcomes. Every time there is an issue or the data set changes, feature extraction is a difficult, time-consuming procedure that must be adjusted[11]. As a result, feature extraction is an expensive endeavor that relies on the knowledge of an expert and is not very generic. Deep learning, on the other hand, does not require feature extraction because it finds the crucial feature automatically during the training phase [12].

Overall, previous reviews show that deep learning algorithms are a promising technique with excellent accuracy. Despite this, the standard of the data utilized in the CNN model is quite important. Depending on how intricate the subject being researched is, at least hundreds of images may be needed [13]. Because of weight sharing and other factors, deep learning algorithms can learn complicated problems quickly. The use of complex models allows for significant parallelization [14].

Only a few attempts have been made to use deep learning techniques to develop a wheat rust classification model using CNN [15, 16].A few initiatives have also been taken to use deep learning to create a wheat-yellow rust classification model. They also lack image preprocessing before use. Therefore, deep learning algorithms were used in this thesis for categorization and

feature extraction to fill in the above gaps. The deep learning technique minimizes (simplifies) domain experts' work by automatically learning high-level features from incremental data sets and avoiding hard feature extraction [17] .

**Research gaps:**

Although there have been some studies on the detection and classification of rust diseases in Ethiopia, there is still a significant research gap in the use of deep learning techniques for this purpose. Most of the existing studies have focused on traditional methods of rust disease detection and classification. There is a need to explore the potential of deep learning techniques for the detection and classification of rust diseases in Ethiopia.

## 1.4 Research questions

The following are the research questions that were attempted in the research:

- What are the suitable hyper parameters that help to enhance the performance of the wheat rust disease detection model?
- What is the impact of different pre-processing techniques (e.g. image resizing, data augmentation) on the accuracy of wheat rust disease detection and classification?
- What are the suitable regularization algorithms for proposed model?
- How to evaluate performance of proposed model?

# 1.4 Objective of the study

## 1.4.1 General objective

The general objective of the investigation is to develop a deep learning model for wheat rust disease detection and classification.

## 1.4.2 Specific objective

To achieve the general objective of the thesis, we formulate the following specific objectives:

- To identify appropriate image processing techniques.
- To design the architecture of the suggested model.
- To identify suitable regularization algorithms.

- To develop a model that uses image processing and classification.
- To train and test the proposed model using the collected dataset
- To identify suitable hyper parameters that help to enhance the performance of the detection model.
- To investigate the impact of different pre-processing techniques, such as image resizing and data augmentation, on the proposed model.
- To assess the model's performance using evaluation metrics.

### 1.2.3 Scope of the study

The scope of this research is to develop a wheat rust disease detection and classification model using deep learning techniques and approaches. The study focuses on three distinct wheat rust infections, including stem rust, yellow rust, and leaf rust. Even though there are other various wheat ailments like powdery mildew, septoria, Alternaria leaf blight, scab (head blight), etc... the thesis does not look at the rest due to the time duration of the study and the lack of quality data and better computational hardwarecomponent like graphics processing unit (GPUs), which are the absolutely crucial resource when working on deep learning. Furthermore, the research data collected from Adet and Arebaminche Agricultural Research Institute Centre does not include other agricultural research centres like Ambo Agricultural Research Centre or Debere Birhan Agricultural Research Centre due to a lack of time and resources.

### 1.2.4 Limitation of the study

The study utilized a dataset containing images of only three types of wheat rust diseases. This constraint may reduce the generalizability of the model to accurately detect and classify other wheat rust diseases not represented in the dataset. The limited dataset may also impact the model's performance in classifying variations of the included diseases that were not covered by the training data. The study was conducted with limited computational resources, which may have constrained the ability to explore more advanced deep learning models that require significant processing power. The limited resources may have also impacted the time required for training and experimentation, restricting the potential for model improvement.

## 1.5 Significance of the study

Artificial intelligence has become a well-known truth that has made human life easier. As a result, it is up to the users to make the most use of it in order to improve their living situations. First and foremost, this research aids pathologists in comprehending the value of artificial intelligence in general as well as machine learning and deep learning in their specific field. Deep learning and machine learning are two of the most popular technologies for detecting wheat rust disease in wheat leaves. Agriculture professionals are able to grasp the value of image processing in the agricultural sector as a result of this research. As a result, image processing allows professionals to quickly diagnose wheat disease. This research also aids farmers in lowering their production costs, which have skyrocketed due to the excessive fungicide use on their plants. Once more, the study's findings provide essential recommendations and comments for the relevant authorities, allowing them to implement necessary agricultural production measures in the event of a crop disease outbreak. Finally, this work will serve as a resource for other academics who wish to do additional research into the issues surrounding plant disease detection.

## 1.6 methods

This section discusses various techniques or tools used to analyze data. These methods can include data preprocessing, model training, and evaluation. Specific deep learning research methods might include data augmentation, learning rate scheduling, and regularization techniques, among others.

### 1.6.1 Literature review

To have a deeper understanding of the topic, many types of literature on wheat rust disease, image processing, and machine learning are reviewed. This enables us to summarize relevant publications that are pertinent to this study. Interviews were conducted with specialists in the botanical field, such as specialized plant pathologists, who should have a thorough understanding of the various diseases that impact wheat.

### 1.6.2 Data collection

We collected sample wheat plant leaves from the wheat-growing regions in Ethiopia.

### 1.6.3 Modeldevelopment

The construction of a prototype for deep learning and digital image processing is being done to address the underlying issues and test the proposed solution. To create the prototype, Anaconda, Python, TensorFlow, Keras, etc. were used.

### 1.6.4 Evaluation

Analysis and appropriate metrics are used to evaluate the proposed thesis project.

## 1.7 Organization of the thesis

The first chapter introduces the thesis paper's background concepts and ideas. A survey of the literature on plant diseases, image processing, machine learning, algorithms, deep learning, and related topics is presented in the second chapter. In chapter 3, the design of the proposed solution, methodologies, and methods, including data collection, materials, tools used in conducting this thesis, selection of an appropriate system, and assessment metrics, are presented. The application of the model as well as the analysis and assessment of the research experiment's findings are covered in Chapter 4. Chapter Five uses recommendations and conclusions to generalize the premise.

# CHAPTER TWO

# 2. LITERATURE REVIEW

## 2.1 Agricultural Disease

Plant diseases and pests are caused by pathogenic agents such as fungus, viruses, and bacteria, as well as living organisms such as insects, rats, cockroaches and fleas, for example, can contaminate food and spread sickness to humans. These disorders are susceptible to biotic and abiotic stressors, which are listed below.

**Biotic factors:** These are living entities that affect the ecosystem, such as animals, plants, fungi, bacteria, and so on. They are capable of spreading from one plant to another and may affect many kinds of leaves, stems, crowns, roots, fruits, and seeds. Plant pathogens are what they're called. All living constituents of an ecosystem provide biotic services.

**Abaiotic factors:**Examples of biotic influences include forest fires, water, the atmosphere, weather, soil, minerals, and human activities. Although these infections cannot be transferred from one plant to another, their impact on plant health can be observed. Abiotic variables refer to all nonliving elements in a given environment, such as the temperature ranges required by different crops, pH levels, and humidity levels [18].

## 2.2 Wheat rust disease

The principal biotic restrictions to wheat production in Ethiopia are wheat rust on the stem and stripes of rust on wheat. In recent decades, principal epidemics have occurred, including a significant wheat stripes rust outbreak in 2010 and a significant epidemic of wheat stripes rust in 2013, both of which resulted in crop losses of as much as one hundred percent and an atypical loss of approximately fifty percent [2]. One of the largest and most popular grain crops worldwide is wheat. It originates from a species of grass called Trinity, which is cultivated in many different forms all over the world. Wheat, in general, is the second-most important product for nutrition, demanding thousands of tons of imports and millions of dollars in real currency in Ethiopia [19]. Despite the crop's importance, it is the agricultural product that is most frequently disease-infected in Ethiopia, with the three prevalent rusts being stem rust, yellow (stripe) rust, and leaf rust.

### 2.2.1 Leaf rust

It is the fungus that causes leaf rust, also known as brown rust (Puccinia triticina). The rust disease can be found in wheat, barley, and many more cereals. The rust on leaves is a fungus that only affects the leaves.

*Figure2.1 sampleimage for leaf rust disease*

### 2.2.2 Yellow (stripe) rust

One of the more dangerous fungal diseases affecting wheat is stripe rust [2]. It affects cultivars of vulnerable hosts, and symptoms start to show around a week after the initial infection. Stripe rust can arise and spread quickly when climatic conditions favor the illness, which frequently results in substantial grain yield and quality loss. Early in the growing period, when temperatures range between two and fifteen degrees Celsius, the disease is most common, but it can also occur at temperatures as high as 24 °C (74 °F). Despite the fact that the leaf and yellow rusts are both classified as members of the leaf rust species, they are separate races that can be difficult to distinguish by visual inspection, necessitating laboratory testing.

*Figure 2.2 sample image for yellow rust disease*

### 2.2.3 Stem rust

Stem rust is a hazardous infection that negatively impacts wheat crops and is brought on by the fungus, like other rust forms (Puccinia graminis) [2]. Barley, triticale, wheat with durum, and

wheat for bread are among the crop species impacted by the disease. Cereal farming has been afflicted by these diseases for millennia.

*Figure 2.3 sample image for stem rust disease*

## 2.3 Image processing

The processing of images entails changing a image  using a variety of techniques in an attempt to improve it and obtain relevant information from it. Image processing enhances the visual look of images and makes extracting important data for further image analysis easier. Image processing in imaging science refers to the application of mathematical operations to the processing of images using any type of signal processing, where the input is a image, series of images, or video, such as a image or video frame, and the output can either be a image or a collection of parameters or image-related characteristics. Most image-processing techniques separate each color plane in a image , treat it like an analogous two-dimensional signal, and then deal with the signal using customary signal-processing techniques. Images are also transformed into three-dimensional signals using either time or the z-axis as the dimension. The use of digital imaging is the most frequent method; however, visual and analogue image processing are also possible [20].

### 2.3.1 Digital image processing

"Digital image processing" is the technique of editing digital images on a machine. Despite being a subset of signals and systems, it focuses on images. DIP's primary goal is to build an image-processing computer system. The system accepts a digital image as input, uses effective algorithms to process it, and produces an output. Adobe Photoshop is the illustration software that is most commonly used. One of the most popular programmes for editing digital images is this one. It used to be only possible for a very small group of specialists with access to expensive equipment to process digital images using computers. Now, nevertheless, the proliferation of digital pictures has led to some people using computerized image processing as frequently as data processing due to

the mixture of powerful machines on every desktop and the undeniable fact that almost everyone has some sort of device for acquiring digital images, whether it be a smartphone, a camera, or scanners [21].

Digital images are interpreted by computers as 2D or 3D matrix structures, with each value or pixel representing amplitude that is sometimes known as the "intensity" of the image. The range of the amplitude value in 8-bit images, which is the standard for us to function with, is 0 to 255.

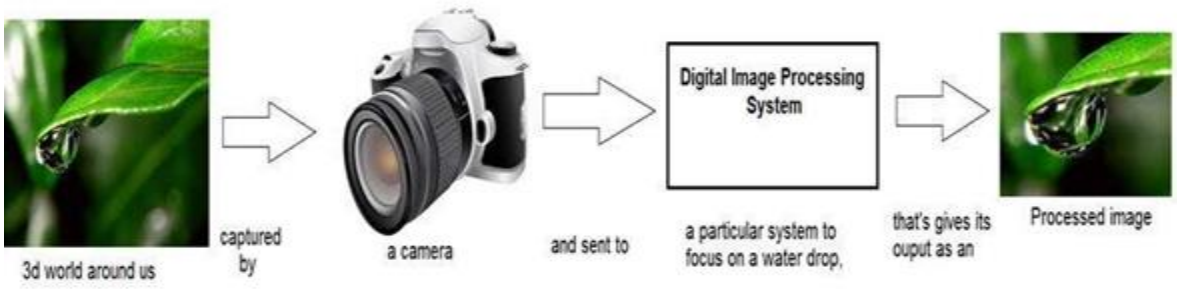



*Figure2.4–working of digital image processing*

In the image above, an analogue image was taken, transferred to a computer system, and then zoomed in such a way as to keep the image quality while removing all but the most important features to focus on the droplet of water.

There are many uses for image processing, including to subjectively enhance an image's quality, typically by boosting contrast. It's known as image enhancement. Because it represents the image with the fewest number of bits possible while maintaining the image's quality. This process is known as image enhancement" and is used to objectively enhance an image, such as by cutting down on blur.

From a handful of specialist applications, image processing has grown and is now fast expanding into a common scientific tool. Today, nearly all fields of the natural sciences and technology use image processing techniques.

### 2.3.2 Basic steps in digital image processing

It takes more than one step to process an image. We are able to discriminate between various procedures that need to be carried out sequentially before we can generate the desired information from the image.

**Image Accusation**

The method of acquiring an image from a source-often one that is hardware-based-so that it can be transported through the subsequent processing processes is known as image collection. The captured image is completely unedited and is the result of whichever equipment was employed to produce it.

One of the most important goals of this strategy is to have a source of input that functions within constraints that are so precise and controlled that the same image can, if required, be almost precisely replicated under identical circumstances, making it simpler to spot and eliminate anomalous factors[22].

**Image Preprocessing**

Image Capture and Preprocessing of Images the many initial picture enhancement methods used on the recorded raw images are referred to as image preprocessing. It is one of the low-level image processing procedures. There is a chance that we will end up with noisy images during image acquisition. Images can have noise from a variety of sources. To create a digital image, the images should first be transferred from the camera to a computer. Digital images, despite what they appear to be on the screen, are in fact only digits that may be read by a computer and converted into tiny dots or other picture components that reflect real objects. Use a scaling manager, which carries out a mathematical change, to change the size of a picture (or a specific area of an object).

Image reductions, or sub-samples, are the technique of substituting an assortment of pixels with one randomly chosen value of pixels from within the group or by interpolating between pixels in nearby regions. To zoom in on an image, pixel reproduction, also known as interpolation, is used. In a multiple-phase image processing chain, scaling can be used as a low-level preprocessor that works on the characteristics of a particular scale to alter an image's visual look, the quantity of data contained in a scene, or both. Scale is an example of an affine change[23].

**Feature extraction**

The primary step in digital image processing after segmentation is feature extraction. Objects must be described in order to be transformed into a format that computers can process. An image feature is the name of this structure. An identifying parameter, trait, or attribute of an image is called an image feature. Major significance is given to image features in image categorization. For image classification, a variety of image feature types have been developed. Some of the major aspects of an image include its morphology, color, and texture. These picture attributes are fed into a machine that classifies things into various groups.

Morphological features, including shape and size, are geometric characteristics of an image. They are actual physical dimensions that define how something appears. For instance, circularity assesses the form of the picture's compactness, while area and perimeter are some of the most frequently assessed size parameters. One of the most commonly used features for categorizing images is color. The brightness of the corresponding location in the image is frequently recorded as a numeric number in each pixel of an image [24].

A combination of a few of these numbers can be utilized to convey color information.This is supported by the statistical significance levels of color parameters, such as average, mode, deviation from the mean, etc., that are used in image systems for categorization. Color is one of the most widely utilized characteristics for classifying images. In many cases, each pixel of a image contains an integer that represents the brightness of the associated area in the image. A combination of a few of these numbers can be utilized to convey the color data. The most frequent brightness value range is 0 to 255 (8-bit range), although an extensive range of 10 or more bits, potentially even 16 (0 to 65,535), may be observed based on the camera's, scanning devices, or other collection device type. Nevertheless, these images are still routinely stored in a succession of discontinuous arrays since it is easier to work with them as well as translate them to displays [25].

**Classification**

The stratification of an image into different classes with related features is made easier by the ability to identify the attributes of objects in an image from a certain set of measured values of the object's features. The major objective of classifier design is to generate a classification label or value that accurately represents the item's class assignment by using certain attributes of the object as inputs. The learning set, a collection of items with predetermined labels or values, serves as the immediate building block for this method. For the recognition of patterns, a variety of pattern

categorization techniques have been employed. The two major types of classification methods are supervised and unsupervised learning. In supervised classification, a sizable collection of labeled training pattern samples is used to train the classifier. Labeled pattern samples refer to a collection of patterns whose class memberships are predetermined [26].

## 2.3. 3Applications Digital image processing

Is an essential part of several phases of preprocessing, such as recognizing faces, recognizing objects, and image reductions. Processing imagery is used to enhance existing images or draw out important data from them. This is critical for several visual systems that utilize neural networks because preparation can dramatically enhance the accuracy of models. In a variety of industries, including photography, design, marketing, advertising, and more, digital image processing is widely employed. Digital image processing has numerous widespread uses in the medical industry, including X-ray imaging, and UV imaging. Remote sensing in space entails using satellites to survey the earth and take note of any spacecraft movements. Another usage of digital image processing is machine vision, sometimes known as robot vision. Although it takes a lot of time, digital image processing can improve people's quality of life [27].

Plant disease classification is one of the most popular uses of image processing. Traditionally, an expert would be consulted for this purpose. Fortunately, in this case, image processing technology can come to the rescue. Digital media is first preprocessed to increase resolution, reduce noise, and enhance color. The image will be improved, segmented, and linked to relevant images in a database after that. The segmented image is then compared to a reference image for the processor to check if it has any flaws. Changing images by adding or removing objects is a different application, especially in the entertainment industry[28].

## 2.4 Computer Vision

The innovations that allow machines to understand images are generally referred to as "vision for computers" technology. This is most typically applied to image identification, a technique for identifying things and their visual features. Face recognition, surveillance of security, structure verification, imaging for medicine, brand and logotype identification, and many other uses of visual identification are currently in use. But the images need to be processed through the labeling, division, or other processing techniques mentioned earlier for these representations to work[29].Some of the most notable use cases for computer vision applications in the present day

are described below.

### 2.4.1 Image categorization

The practice of classifying an image based on the surrounding visual information is called as image categorization. The process involves focusing on the connections between nearby pixels. The classification system consists of a database with predetermined patterns.

To classify the recognized object, these patterns are contrasted with the object. Image classification is useful in a variety of applications, including biometry, video surveillance, vehicle navigation, and biomedical imaging [30].

### 2.4.2 Agriculture

To monitor crops and address issues like weed growth and fertilizer insufficiency, several farming companies use vision technology. Computerized vision systems analyze satellite, drone, and aircraft imagery to spot problems quickly and avert losses in revenue[31].

### 2.4.3 Objects Recognition

Finding one or more objects in an image or video is the focus of this area of computer vision and AI [32]. For instance, surveillance cameras can intelligently identify people and their actions (such as inactivity, the presence of weapons like knives or firearms, etc…) and flag those as suspicious.

### 2.4.4Autonomous Vehicles

Prior to being approved for commercial use, the most advanced autonomous driving technology had not yet realized all of its promise. However, it has proven able to merge images recognition into computer vision algorithms to have pedestrian detection capacity and to stop when a stop sign is displayed.

### 2.4.5 Robotics

Numerous robotics-based initiatives have used image recognition to teach robots to recognize objects for improved navigation and to find objects that may be in their way.

### 2.4.6 Text recognition

Another interesting addition made with the use of picture recognition is text detection. Text detection has the potential to identify text and characters from an image, such as a photograph that may contain a street sign or a traffic sign. One of the leading firms in text identification is Cloud

Vision from Google.

### 2.4.7Facial identification

Facial recognition has been a possibility since the advent of AI. Due to its potential, face recognition is in high demand in the industry for both surveillance and device security. Several experts, though, are raising concerns about the technology's privacy features. The reality remains, nevertheless, that every technology has significant drawbacks. As a result, the effective application of facial recognition algorithms will lead to the development of necessities of life like traffic and city monitoring.

### 2.4.8 E-Commerce

Customers can now upload images of goods they already own or goods they want to find complementing styles for in order to search for comparable goods. This calls for converting the image into a visual embedding, where the suggested items are either those that are similar to the one that was uploaded or those that are known to be complementary.

The process of applying algorithms to images is known as image processing. These algorithms frequently aim to enhance the image's quality or modify it for a distinct visual appearance. To prepare images for computer vision models, image processing is also crucial. This includes applying segmentation or tagging recognized items [33].

The innovations that allow machines to understand images are generally referred to as "vision for computers" technologies. This is most typically applied to image identification, a technique for identifying objects and their visual features. Recognition of faces, surveillance of security, architectural recognition, imaging for medicine, brand and logotype recognition, and many other uses of image recognition are currently in use. But before the models can work, the images must go through labeling, referred to as "vision for computers" technologies.

## 2. 5 Machine learning technique

Experts can examine, comprehend, and explore enormous, complex datasets using machine learning algorithms, which are pieces of computer code (programming logic or arithmetic). Each algorithm adheres to a set of guidelines to accomplish the task of classifying data or producing predictions by discovering patterns concealed in the data [34].

Algorithms used for machine learning describe the steps and ideas that a system should consider while resolving a certain problem. These algorithms replicate the situation and examine the data to predict the result within a certain range. These mathematical models also learn from previous performance in outcome prediction and adjust, optimize, and improve when new data is added. Machine learning algorithms, to put it simply, tend to become "smarter" with time.

Using a variety of variables, including the type of algorithm employed, machine learning models analyses data and produce precise results utilizing a range of factors. A smaller dataset used for training purposes produced these characteristics.

### 2.5.1 Types of machine learning algorithms

The machine-learning techniques are categorized into four groups based on the learning methodologies: supervised, semi-supervised, unsupervised, and reinforcement learning. Regression and classification algorithms are the most popular methods for forecasting values, identifying shared characteristics, and discovering unusual trends in datasets[35].

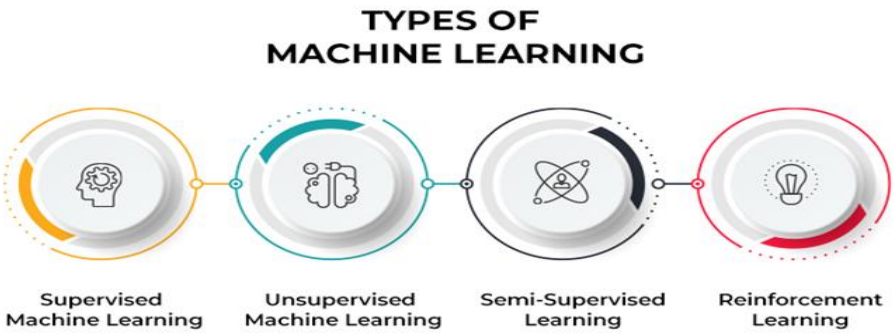



*Figure2.5typesofmachinelearningalgorithms*

## 2.5.1.1 Supervised learning

Using supervised training and the concept of functional estimation, we train a method and then select the function of the algorithm that best captures the input data-that is, the function that, given a given X, best predicts the value of y (X -> y) at the end of the procedure. The method relies on a human assumption about how the computer should learn, and this assumption introduce a bias, which is why we are frequently unable to identify the real function that always makes the proper predictions[35].

We feed training information, including inputs and predictions, to the machine while simultaneously displaying the correct solutions (output). A machine ought to be able to recognize patterns in the data. Here, teachers are human experts. They seek to model the connections and interactions among the objective predictions output and the input features in order to predict the resultant parameters for fresh data based on the connections that the supervised learning methods have learned from the prior data sets. Predictions are made by supervised learning systems using labeled datasets. This method of learning is useful when you know the kind of output you want.

Decision trees, logistic regression, linear regression, support vector machines, and neural networks were among the most well-known supervised machine learning methods [35].

**Artificial neural network**

Artificial neural networks behave similarly to neurons in our nervous system. The word "neural" derives from "neuron," or nerve cells, which make up the basic functional unit of the human (and animal) nervous system and are found in the brain and other areas of the body. A neural network is a collection of algorithms that validate the fundamental connections in a dataset resembling the human brain. Without altering the output process, the neural network helps to modify the input such that the network produces optimal results [36].An artificial intelligence technique called a neural network instructs computers to analyses data in a manner modeled after the human brain. Deep learning is a sort of machine learning that employs interconnected neurons or nodes in a layered framework to mimic the human brain. It develops an adaptive system that computers utilize to continuously learn from their errors and improve. Artificial neural networks, therefore, try to find solutions to difficult challenges.

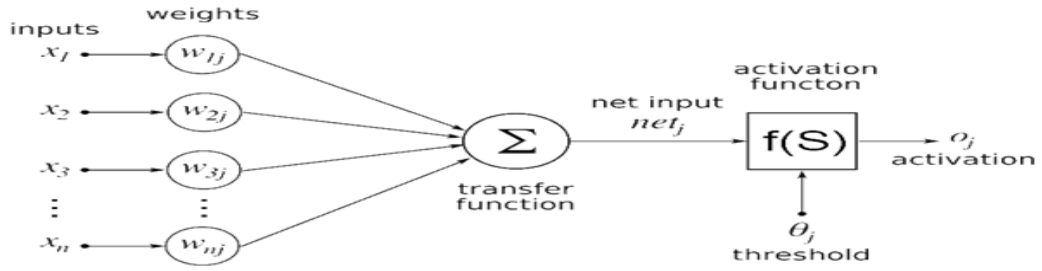



*Figure 2.6workingofartificialneuralnetworks*

The strength of the relationships among neuronal cells is typically represented by the weights that are used. To acquire the required result for the issue that was defined, the function that activates the problem is a transfer function that is applied. There are numerous activation mechanisms. To mention a few: Softmax, Bipolar, Binary Sigmoid, Bipolar Sigmoid, Hyperbolic Tangent, Sigmoid Hyperbolic, Linear Regression, Logistic Regression Identification Function, and Rectified Linear Unit (ReLU) Activation Function Using methods of learning, artificial neural network systems are specifically created for specific tasks, including binary classification, multi-class classification, pattern recognition, and more. With study, the weights of the synaptic connections between the two brain networks change[37].

**Support vector machine**

A support vector machine (SVM) is a supervised machine learning method that divides data into two categories and then performs classification or regression tasks using that division. Both classification and regression tasks are carried out using support vector machine methods. Because each feature value has a corresponding coordinate value, plotting the features is simpler. By clearly identifying the hyper-plane that divides the two sets of support vectors, or classes, classification is further carried out. A proper classification between the depicted data points is ensured by a strong separation.

A linear model for classification and regression issues is the support vector machine. It works well for many real-world issues and can solve both linear and non-linear problems. The SVM concept is straightforward: a line or a hyperplane that divides the data into classes is produced by the algorithm[38].

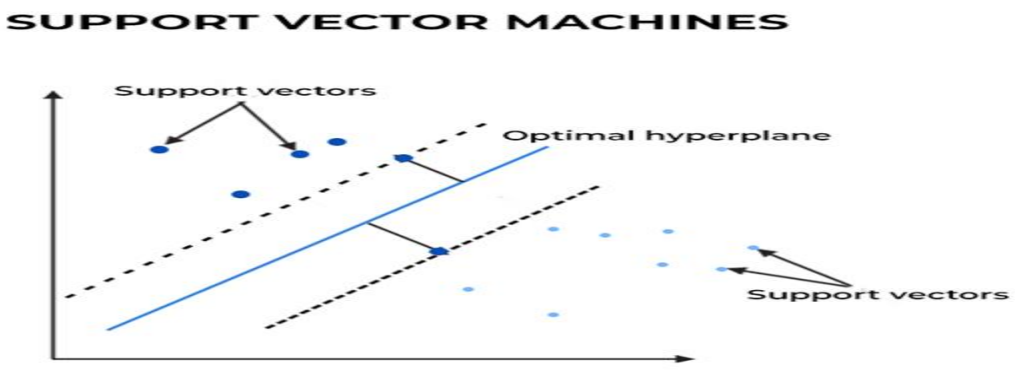


*Figure2.7workingofSVM*

We locate the points from both classes that are closest to the line using the SVM technique. Support vectors are the names for these points. The distance between the line and the support vectors is now calculated. The margin refers to this separation. Maximizing the margin is our aim. The best hyper plane is the one for which the margin is greatest. Using supervised learning models, a support vector machine (SVM) solves challenging classification, regression, and outlier detection issues by carrying out the best possible data transformations to establish boundaries between data points based on predefined classes, labels, or outputs. SVMs, to put it simply, represent the coordinates for distinct observations. These are well-known machine learning classifiers that are used in a variety of apps, including speech detection, steganography detection in digital images, text classification, facial expression classification, and data categorization [39].

**A decision tree**

The algorithms for decision trees may be useful for creating several possibilities for a specific decision as well as for predicting the best option according to a mathematical design. The selection node is the main node of the tree, where the sub-nodes representing various results branch off. Each outcome has the ability to generate new child nodes, which may result in more outcomes. The technique builds an arrangement that mimics a tree to solve problems with categorization[40].

**Logistic regression**

The dependent variable is binaryin the logistic regression model. This style of investigation examines the relationship between a single categorical parameter and a number of separate variables by analyzing the data. In predictive analysis, where pertinent facts forecast an event's probability using a legit function, logistic regression is applied. As a result, it is also known as legit regression[41].

**2.5.1.2Unsupervised learning**

Data without labels is used by autonomous learning techniques. This teaching strategy labels the unidentified material by categorizing it or by expressing its type, form, or structure. This strategy is advantageous when the type of result is unknown. There isn't a single instructor in this scenario;

instead, a machine might be able to educate you when it finds patterns in the data. Since there are no output labels or classifications, the method cannot try to model associations. These computer programmes try to use the information that is provided to look for trends, mine for rules, collect and sum up the data points, identify patterns, and derive pertinent insights that help users comprehend the data better. Algorithms are especially useful when an expert human being is unsure of what to look for in the data. Clustering using K-means, clustering based on hierarchy, and the apriority algorithm are the most popular undetected learning techniques [35].

**K-Means**

Clustering tasks are carried out via the distance-based unsupervised machine learning algorithm K-Means. With the help of this approach, you may group datasets into clusters (K clusters), where the data points from one cluster remain homogeneous and the data points from two distinct clusters remain heterogeneous.

These steps are used to create the clusters under K-Means:

Initialization: The K-means method chooses centroids (a K' number of points) for each cluster.

Assigning items to the centroid: At each data point, clusters are created using the K nearest centroids.

Centroid news: The closest distance for each data point is calculated using new centroids that are based on existing clusters. When necessary, the centroid's position is updated at this location.

Repeat: Continue until the centroids remain the same[42].

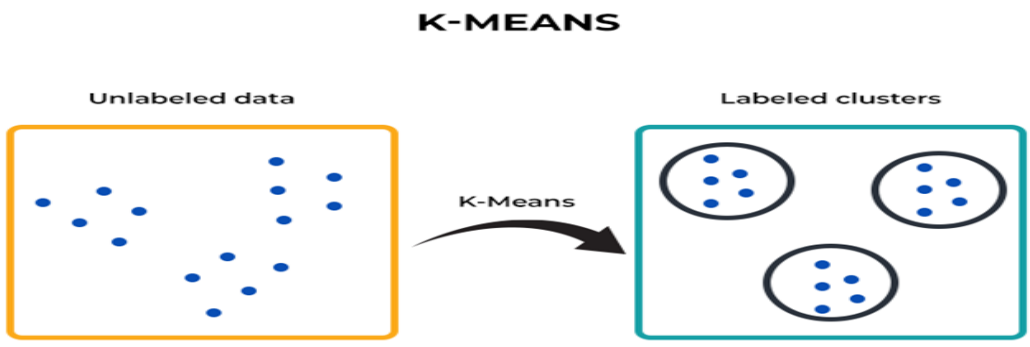



*Figure 2.8 working of k-means algorithms*

**Hierarchical clustering**

Objects are divided into groups according to how comparable they are using a process called hierarchy clustering, also known as the analysis of hierarchical clusters. As a result, a number of clusters are formed, each of which is distinct from the others but nevertheless shares many characteristics. The application of clustering to hierarchy: a hierarchical clustering technique first treats each observation as its own cluster. It then regularly accomplishes the following two tasks: The two clusters that are most similar to one another should be identified first, and then they should be combined. Until all of the groups are merged, this iterative process continues. The illustrations that follow demonstrate how to do this.

Hierarchical clustering is one of the methodologies for clustering that deep learning applications employ the most frequently. Before trying to appreciate the hierarchical clustering method, let's first establish the grouping concept. In its simplest form, grouping is a method for grouping linked points of information into sets that have members that are more comparable to one another than to those of other sets. The term "cluster" refers to the grouping of connected points of information.

### 2.5.1.3 Semi-Supervised Learning

Combining the two methods stated above, semi-supervised machine learning techniques use both labeled and unlabeled data. The objective of these methods is to categories data without labels, employing the understanding gained from labeled data.

In the initial two categories, either each data point in the information set has a category or none of them does. Semi-supervised learning can help in this situation. The cost of labeling is relatively high because it frequently requires engaging seasoned human specialists. Consequently, when labeling is missing from the majority of data but present in a small number of them, semi-supervised approaches are the best choices for model construction. These methods make use of the fact that, although the grouping identities of the unlabeled data are unidentified, this data still provides essential information regarding the group characteristics [43].

#### 2.5.1.4 Reinforcement learning

Techniques for learning through reinforcement use the final product or outcome as a benchmark to choose the next step. In other words, these algorithms decide whether to take the next action or not after studying previous outcomes and getting advice at each stage. The framework determines if it made a good, poor, or neutral judgment during the process. Computer programmers can apply

reinforcement learning since they are designed to make judgments with little or no input from humans [35].

Utilizing information gained through encounters with the surroundings, the reinforcement education strategy seeks to achieve the greatest reward or minimize risk. In a method for reinforcement learning, a learning agent continuously and repeatedly learns from its environment. The agent slowly learns from its encounters with its surroundings until it has looked into every possible state.

Reinforcement learning, a type of machine learning, is also an aspect of artificial intelligence. It enables both software and hardware agents to instinctively choose the behavior that produce the greatest efficiency under a specific circumstance. A simple incentive response message, or reinforcement signal, is required for the agent in question to learn its action[35].

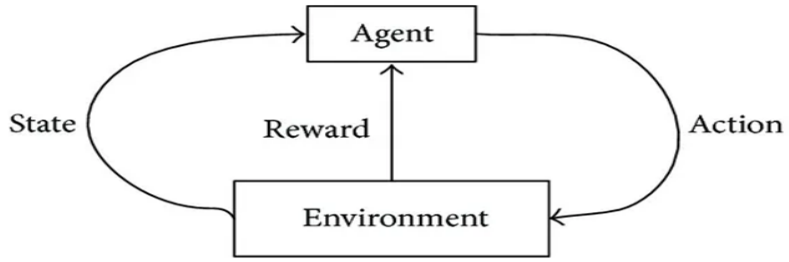



*Figure 2.9Reinforcementlearning*

In order to develop intelligent algorithms (sometimes referred to as agents), learning through reinforcement involves three phases:

- The programmer pays attention to its input state.
- The ability to make choices mechanism is used to compel the agent to act.
- Following the activity, the surroundings encourage or compensate the participant.
- Data concerning the state-action pair of the award is kept.

## 2.6 Activation Functions

A neuron's activation status is determined by a mechanism of activation. By employing simple mathematical techniques, it will be determined whether or not the neuron's input to the neural network is significant during the procedure of prediction. The node's (or layer's) activating function's job is to generate outputs given a set of input variables [42]. Usually,

a separate activation mechanism than the rest is used for the last fully connected layer. A mechanism for activation has to be selected in accordance with the requirements of every assignment. For the multiclass categorization assignment, a soft-max function is employed as an activation function to normalize the output actual numbers from the final fully connected layer to target category probability, where every number varies from 0 to 1, so they collectively add to a maximum of 1. In this inquiry, Soft-Max is utilized [44].There are two primary groups in which activation operates

### 2.6.1 Linear activation function

As we observe, the operation is linear in the form of a function of activation. Consequently, nothing is going to be employed to restrict the operations' outputs.

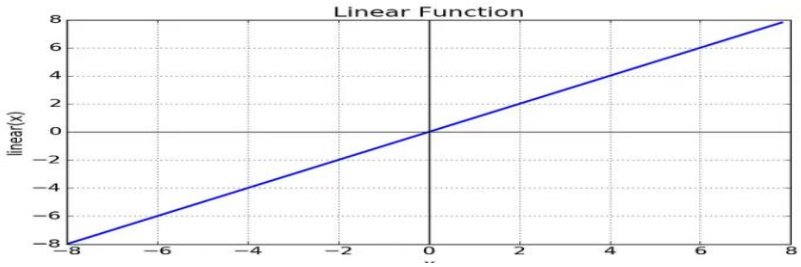



*Figure 2. 1 Linear activation function*

### 2.6.2 Non-linear activation function

Nonlinear activation processes are among the more widely utilized for activation purposes. According to nonlinear behavior, the data structure looks like this: It promotes outcome separation and the generalization or adaptation of the framework to a range of information [44].

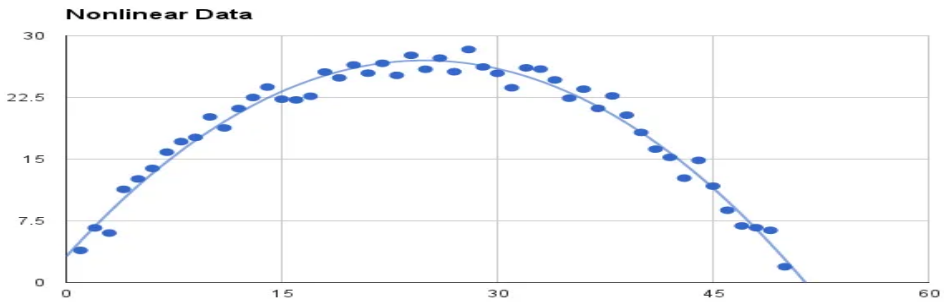



*Figure2.11 non linear activation function[44]*

**Softmax Activation Function**Softmax is yet another form of enactment function utilized in neural processing. A probability estimate is produced using a set of actual figures using this technique. The result of the Softmax task is a quality spectrum of 0 to 1, with a total probability of 1. The Softmax work returns probabilities for each class in multi-class models, with the objective class having the highest probability. Practically every yield layer in frameworks for deep learning makes use of Softmax's work.A set of K real numbers is normalized by the Softmax function into a probability distribution such that their total is 1. Softmax is a soft variation of the max() method, as the name would imply.

$$S(z) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_i}}$$

softmax function

TheSoftmax function and the sigmoid function are interchangeable for K = 2. In general, binary classification issues are solved using the sigmoid function, and multi-class classification issues are solved using the softmax function[45].

**Sigmoid activation function** - The mathematical function known as the "sigmoid activation function" has an easily identifiable "S" curve. It is used for the implementation of fundamental neural networks and logistic regression. The sigmoid function is the best option if we want a classifier to solve a problem with more than one correct response. This function should be independently applied to each component of the raw output. The sigmoid function often returns values between zero and one, or between -1 and 1. A hyperbolic tangent function has also been applied to artificial neurons in addition to logic. It has also been applied as a cumulative distribution function in addition to this. It is simple and cuts down on the amount of time needed for installation. Derivatives have a small range, which results in a large loss of information, which is a severe negative.

The sigmoid function appears as follows:

$$f(x) = sigmoid(x) = \frac{1}{1 + e^{-x}}$$

The more layers in our neural network, the more data is compressed and lost per layer, amplifying the overall data loss and making it substantial.

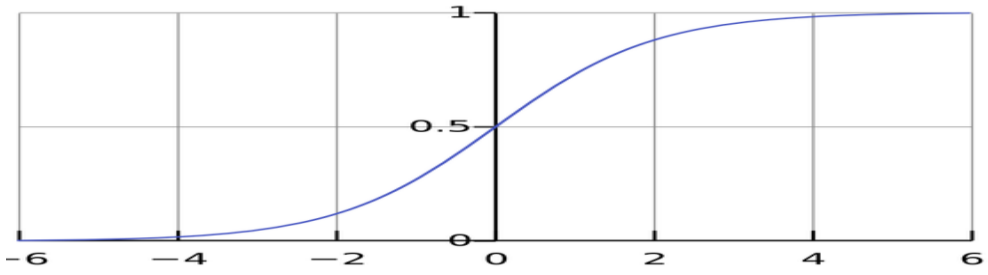



*Figure 2.12 how the sigmoid function looks like[46]*

**ReLU(Rectified Linear Units) Activation Function**

The ReLU feature is currently the mechanism for activation that is used most frequently worldwide. Because nearly every machine learning algorithm and convolutional neural network (CNN) uses it.

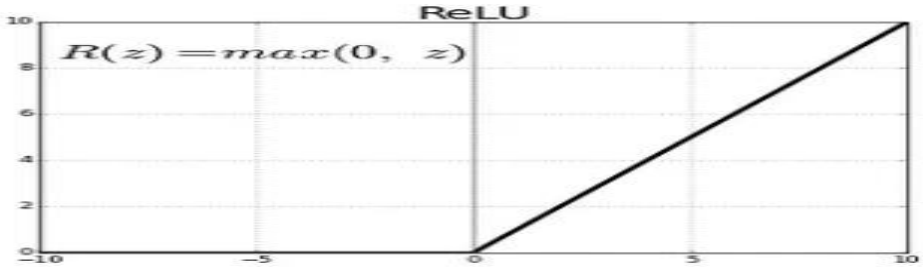



*figure 2.13 how Relu activation function looks like*

As shown, the ReLU has been repaired in half, starting at the bottom. f(z) becomes zero when z is below zero and matches z whenever z is greater than or closer to 0.

Range: [from zero to infinity]

Both the derivative of the value and the function themselves have monotonous behavior. As a consequence, the plot of any unfavorable input to the function that activates ReLU instantly changes to zero, which has an impact on the final graph because the zeros are not properly mapped. The issue is the fact that any unfavorable numbers instantly turn into zero, making it harder for the algorithm to accurately fit or train on the information at hand[47].

**Traning Networks-**System training involves identifying weights for layers that are completely connected and kernels for convolution layers on a training collection that minimize discrepancies between output predictions and given ground-truth labels. The back propagation algorithm, a widely used method to develop artificial neural networks, depends critically on the loss function and the gradient descent optimization procedure. An optimization technique, such as back-propagation and gradient descent, among others, updates a learnable parameter, such as cores and weights, in line with the loss value. Through the use of advanced propagation on a training dataset, a loss function determines how well a model performs with particular units and inputs.

**Loss function** - The degree of agreement between the output predictions made by the network through forwarding propagation and the provided ground truth labels is measured by a loss function, also known as a cost function. For data that is continuously collected, the average squared error is widely employed, while cross-entropy is a popular loss function for multiclass classification. The type of function that loses is one of the parameters that have to be selected according to the assignments given.

**Over fitting**-When a model acquires statistical trends specific to the set used for training, memorizing the unimportant noise instead of the message, and then behaves badly on another set of data following that, the model is considered to be over fit. An over fitted model can't be used to analyses data that has never been seen before, which constitutes one of the core problems with neural networks. A collection of tests is essential to completing a precise assessment of artificial intelligence algorithms in this respect, as was discussed in the preceding section[48].

## 2.7 Deep learning technique

A subtype of machine learning called "deep learning" has more strength and adaptability than conventional machine learning techniques. The key advantage of deep learning techniques is that they utilise an effort to gradually acquire intricate characteristics through information. Due to this, classical machine learning no longer requires manual feature extraction, which requires human involvement[49].Deep learning uses unstructured data and simulates the functioning of the human brain, making it particularly useful for applications like speech and image recognition. When we have a vast amount of data and strong computing performance, it performs more and more. When it comes to challenging issues like speech detection, NLP, and image classification[50].It performs better than alternative strategies. For deep learning, we require two essential components. The first

is having a lot of data to train our model with, and the second is having a good computing device. Deep learning models can learn up to thousands of parameters; therefore, our equipment needs to be able to handle those enormous groups of parameters with ease.The field of deep learning is a branch of artificial intelligence that relies on information representation techniques instead of specific job techniques and uses a neural network as its framework. Artificially networked applications have increased more quickly than ever in the past ten years because of powerful machines (including affordable processing units like GPUs) and a large amount of data. An artificial neural network has one or more phases of processing. The issue we're trying to resolve dictates how many layers of network infrastructure you use. A network of nodes with a deep design has typically have two or three layers at most[51].

### 2.7.1 Convolutional Neural Networks

Processing data with a grid-like structure, such as images, typically makes use of the CNN class of deep learning models. CNN was developed to autonomously and continually learn the geographical hierarchy of characteristics, from low-level to high-level patterns, and was inspired by how an animal's brain's visual cortex is structured. A typical CNN is composed of three different types of layers: convolution, pooling, and fully linked (dense) layers. The first two layers-convolution and pooling-carry out the extraction of characteristics; the third layer, which is fully connected, transforms the features that are extracted towards the results, including categorization. An important part of convolutional neural networks that consists of an accumulation of computations is a convolution layer, a particular form of linear function [51].

In digital images, the values of pixels are recorded in a two-dimensional (2D) grid, which is made up of a kernel-sized grid of parameters and an array of integers. Then, every image point receives the application of an optimizable feature extractor. Since a characteristic can appear anywhere in the image, this is quite useful for image manipulation. As one layer feeds its output onto the next, the characteristics extracted can build systematically and become ever more complex. In training, variables like kernels are optimized to close the gap between outputs and labeled real-world data[18].

There are two main parts to the convocational neural network architecture.

Feature extraction is the technique by which the convolution tool distinguishes between the distinct characteristics of the image for examination. The layer that is completely connected, which makes use of the results of the convolution procedure's results, determines the image's class according to the features that have been previously retrieved [52].

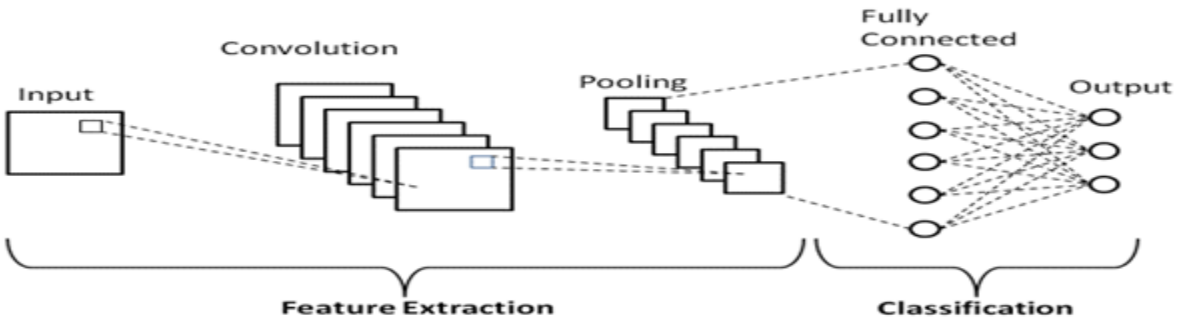



*Figure2.14CNNarchitecture*

### 1 Convulation Layer

Three main types of levels make up the CNN: fully connected (FC), grouping, and convolution layers. When all of these components are combined, the CNN architecture is produced. Along with the aforementioned three distinct layers, two more important factors are the dropout layer and the activation function, which are described below.

The process of convolution of structures a convolution layer, which is responsible for the extraction of features, is the foundation of the CNN architecture. Convolution procedures and activation functions, for example, are examples of linear and nonlinear procedures that are frequently combined to obtain features. The convergence process allows the central point of every kernel to overlap with the outermost portion of the tensor being processed, reducing the height and width of the output feature map relative to the original tensor. Cushioning, usually zero padding, is a technique that is used to address this issue. It requires adding columns and rows of zeros to the tensor of the input on either side in order to fit the kernel's center on the outermost element while keeping the same in-plane dimensions that range throughout the convolution process. Contemporary CNN designs often use minimal padding to retain

in-plane efficiency. A stride is the separation from the following kernel indications and also serves as a description of the convolution procedure. While a stride length of 1 is usually used, it is sometimes possible to down-sample the characteristic mappings by utilizing a stride above one. An alternative approach to achieving down sampling is a sharing procedure, as described below. The essential feature of a convolution procedure is that cores are utilized throughout every one of the image locations; this layer is used to extract the various features from the input images initially. At this level, a mathematical operation called convolution is performed among the input picture and a filter with the dimensions M*M. By sliding the filter over the input picture (M*M), the dot product that exists between the filter's individual parts and those of the input image is calculated. The outcome, dubbed the "feature map," offers information regarding the image, such as its corners and edges. Later, additional layers are given access to this feature graph so they can learn fresh characteristics from the image being used.

**2 Pooling layer**

Frequently, a convolution layer is followed by a layer for pooling. The major objective of this layer is to reduce the size of the convolved feature map in order to reduce computational costs. This is achieved by minimizing the connections between layers and working independently on every element map. Regardless of the approach, different pooling techniques exist. The characteristic map provides the largest contribution to maximum pooling. With the method of average pooling, the components of a segment of a picture with established dimensions are averaged out. The total number of elements within a specified segment is calculated using pooling. Usually, the FC Layer and the Convolution Layer are connected through the Combination Layer. Stacking Pools By applying a pooling layer's typical reduction technique, which introduces translation invariance and minor changes in the learnable variables, the in-plane dimensionality of the map of features, is lowered. Although filter size, stride, and padding are hyper parameters in pooling operations, similar to convolution processes, it is significant to note that neither of the pooling layers contains an accessible variable.

**3 Fully connected layer**

The fully connected (FC) layer, which also has biased and weighted connections,is used to link neuronal cells across the two different layers. These components are often positioned before the output layer, so they make up the final few components of a CNN design. The input image from the layers above is flattened during this procedure and fed to the FC layer. After that, the compressed vector goes through several more FC sections, where actions on the functions of

mathematics frequently take place. At this moment, the method of classification begins taking action. entire linked layer The output feature maps from the last convolution or pooling layer are typically flattened or transformed into a one-dimensional (1D) array of numbers (or vectors) and connected to one or more thick layers, also known as fully connected layers, where every input and every result are linked together by an accessible weight. The characteristics produced by the convolution layers and the down sampling layers are then mapped to the network's final outputs, such as the probabilities for each class in the classification tasks performed by a portion of the fully connected layers. In the final, fully connected layer, the number of nodes that are output often corresponds to the total number of categories. Each fully connected layer is followed by an irregular layer.

## 2.8 RelatedWork

Detection and Classification of Coffee Leaf Disease using Deep Learning [53] The developed a deep learning model that was trained using a set of 1,120 images from the Wolaita Sodo agricultural research centres. An augmentation method was also used to address the issue of information overfitting, resulting in a total of 3,360 pictures. They compared training from scratch and transfer learning strategies to get the best outcomes when classifying such conditions. As a result, training from scratch performs at a rate of 98.5%, whereas transfer-based learning gives correctness rates of 97.01% and 99.89% when using transfer learning through Mobilnet and Resnet50, respectively. Pictures may be classified more accurately with the pre-trained Resnet50 model than with other techniques. By incorporating additional data along with the existing pre-trained models, we are advancing our efforts toward taking into account the second category of Coffee Leaf Disease.

Development of a chickpea disease detection and classification model using deep learning[54]In this study, a model for the identification of chickpea disease was created utilizing deep learning methods, integrating Softmax for classification with Convolutional Neural Networks (CNN) and Long Short-Term Memory (LSTM) for feature extraction. Several picture preprocessing processes, including image resizing, normalization, and noise filtering with a combination of Gaussian filters (GF) and Median filters (MF), were carried out to construct the proposed model. Augmentation was employed to prevent the overfitting issue, and 8391 photos were used to train and test the created model's efficacy. 20% of the information gathered from the captured images

was utilized for testing, 80% of the dataset was used for training, and 80% of the training data was used for validation. With an accuracy of 92.55%, the suggested CNN-LSTM did well in diagnosing chickpea illness. The study's conclusions show that the suggested CNN-LSTM works better than current.

Computer Vision Framework for Wheat Disease Identification and Classification Using Jetson GPU Infrastructure [55]The researcher used a variety of deep learning models, including VGG16/19, Resnet50, and Inceptionv3. Together with the Ethiopian Bishoftu Agricultural Research Institute, this study was carried out. Their main goal was to use sophisticated deep-learning techniques and picture data to automatically identify plant diseases. RGB picture data from the Bishoftu region was gathered for the experiment. Based on the experimental findings, the VGG19 model correctly identified wheat illness in 99.38% of cases.

Plant Disease Detection and Classification by Deep Learning [56]In order to identify and categorize the signs of plant diseases, numerous developed or modified DL architectures are used in conjunction with a number of visualization techniques. Additionally, a number of performance indicators are employed to assess these structures and methodologies. This article offers a thorough justification of the DL models used to depict various plant diseases. Additionally, some research gaps are noted that can be filled to increase transparency for spotting plant illnesses even before their symptoms are visibly apparent.

Ethiopian Coffee Plant Diseases Recognition Based on Imaging and Machine Learning Techniques [57]This article uses image and machine learning approaches to identify the three kinds of coffee illnesses. The Southern Nations, Nationalities, and Peoples, Jimma, and Zegie regions of Ethiopia, where more coffee is grown and data is taken. In this study, autonomous maps (SOM), radial basis functions (RBF), k-Nearest neighbours (KNN), artificial neural networks (ANN), and Naive Bayes are all used. To obtain highly connected and more representative features, we test every set of characteristics. There are 9100 different data sets in total. Out of the total of 9100, 70% were used for testing and the other 30% for training. The overall outcome demonstrated that colour features indicate more than texture aspects when it comes to the

identification of coffee plant illnesses, and the detection rate of the RBF and SOM combined is 90.07%.

Deep Convolutional Neural Network based Detection System for Real-time Corn Plant Disease Recognition [58] This study describes a real-time technique for detecting maize leaf disease based on deep convolutional neural networks. On a computer with a GPU, the efficiency of deep neural networks can be increased by altering the pooling combinations and hyperparameters. The generated model's parameter count has also been optimised to make it appropriate for real-time inference. A dedicated CNN hardware block-equipped Intel Movidius Neural Compute Stick was used to deploy the pre-trained deep CNN model onto a Raspberry Pi 3. The deep learning model obtains an accuracy of 88.46% for identifying maize leaf illnesses, proving the viability of this approach. The provided approach for recognising maize plant diseases can be used by drones and standalone smart devices like the Raspberry Pi or smartphones.

Ethiopian coffee leaf diseases identification using deep learning features [53]This study examines the four main coffee leaf diseases—brown eye spot (BES), coffee berry disease (CBD), coffee leaf rust (CLR), and coffee wilt disease (CWD)—that lower coffee yield in Ethiopia. In this research, researcher suggested a method for identifying Ethiopian coffee leaf diseases using neural network features. The regions of Ethiopia where additional coffee is grown, including Jimma and Zegie, are where the photographs of the coffee leaf illnesses were taken. To eliminate noise from coffee leaf photos, we evaluated gaussian filtering, average filtering, and a hybrid of the two methods of filtering. We found that the combination of the two methods of filtering produced better results. Additionally, we used CNN to extract features and KMeans clustering to segment the data. In the end, we compared the CNN-Softmax classifier to the CNN-SVM classifier. The results of the experiment demonstrated that SVM outperforms softmax classifiers in terms of performance and computational time. A total classification accuracy of 96.5% was attained using our suggested model and SVM classifier.

Detection of rust disease in wheat crops using CNN via transfer teaching [60]. The suggested approach was used to train a deep convolutional neural network (DCNN) to learn to recognize two different rust infections via transfer learning techniques. The application of computer vision models that were trained on sizable datasets is made possible via transfer learning. Examples of DCNN models that were previously pre-trained on large image collections include AlexNet, VGG-16, VGG-19, ResNet-18, ResNet-34, ResNet-50, ResNet-101, and ResNet-152. The DCNN was

evaluated based on 415 photos in a collection. All models' performance on a number of different performance parameters (including accuracy, precision, recall, specificity, MCC, Cohen's Kappa, and FPR) were examined, with ResNet-152 delivering the highest average score. The greatest degree of correctness was achieved at 90.36%[60].

*Table2.1comparison of related works*

| No | **Title** | **Methodology** | Accuracy | Gap | Year |
|---|---|---|---|---|---|
| [54] | Development of a chickpea disease detection and classification model using deep learning | deep learning methods, integrating Softmax for classification with (CNN) and (LSTM) for feature extraction | 92.55 % | Applicable only for chickpea disease, highest accuracy achieved was 92.55% is not enough. | 2022 |
| [55] | Computer Vision Framework for Wheat Disease Identification and Classification Using Jetson GPU Infrastructure | Deep learning | 99.38 % | Researchers only train pre trained models VGG16/19, Resnet50, and Inceptionv3 | 2021 |
| [57] | Ethiopian coffee leaf diseases identification using deep learning features | Deep learning | 96.5% | Study is concentrated on coffee plant | 2021 |
| [60] | Detection of rust disease in wheat crop using convolution neural networks via transfer learning | Deep convolution neural networks (DCNN) have been trained to learn to identify two types of rust disease with the | 90.36 % | Yellow rust disease was not included Small dataset only 415 images, highest accuracy achieved was | 2021 |

| | | | | | |
|---|---|---|---|---|---|
| | | help of transfer learning. | | 90.36% it is not enough | |

# CHAPTER THREE

## 3. RESEARCH METHODOLOGY

The types of problems that are worth looking into, what qualifies as a researchable problem, testable hypotheses, how to frame a problem so that it can be investigated using specific designs and procedures, and how to choose and develop appropriate data collection methods are all covered in research methodologies [68].The framework ofguidelines and procedures known as "experimental research design" was developed to conduct experimental research using a scientific methodology.Experimental research is a study carried out using a scientific methodology that aids a researcher in collecting the information required for better research judgements and carrying out their research objectives with greater clarity and transparency in order to determine the study's findings.

Research methodology is a theory of how an inquiry should proceed and involves the analysis of the assumptions, principles, and procedures of a particular approach to inquiry. It is an action plan, strategy, process, or design that lies behind the selection and use of methods[69]. In this study, image processing is carried out using an experimental research methodology. To choose more efficient algorithms for creating an ideal model, we gather data and run experiments. Preparing data sets, implementing them, and assessing their performance are all parts of applying experimental research[70].

### 3.1 Research design

Research design acts as a link between the formulation of research questions and the execution, or implementation of the research plan. The process of designing an experimental research project contains six parts: Identification of the issue and inspiration for a solution, specification of the goals for a solution, design and development, demonstration, assessment, and communication [71]

### 3.1 Research flow for proposed model

The methodological analysis of developing the wheat rust disease detection includes the following procedures:

- Identifying the domain of the problem.

- Conducting a literature survey on earlier investigations made on plant disease detection and classification by using machine learning and deep learning algorithms
- Problem definition and formulation of objectives, including general objectives and specific objectives,
- Collecting healthy and infected wheat images and dataset preparation by employing various methods like image preprocessing and augmentation.
- Designing the proposed solution to detect the wheat rust disease
- Implementation of the proposed solution with appropriate tools and methods.
- Train and test the suggested model by using an appropriate dataset.
- Performance evaluation of the suggested model.

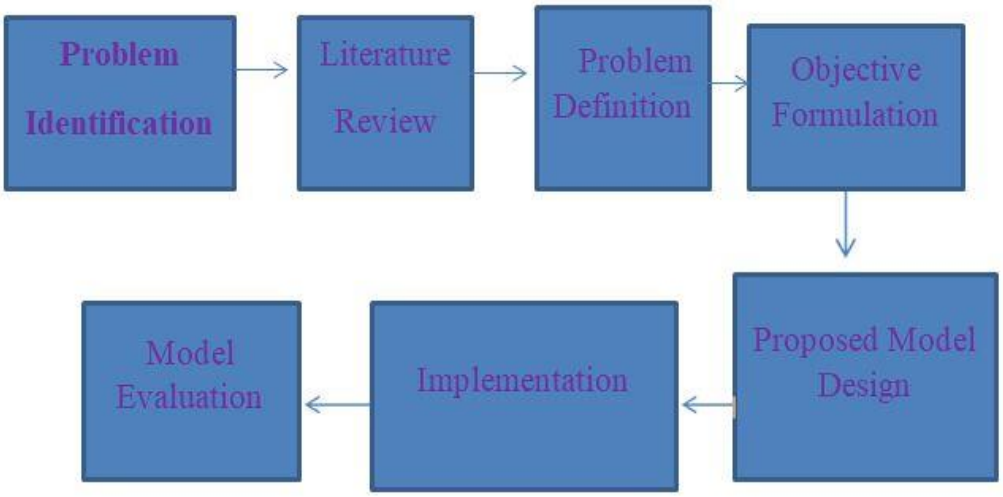



*Figure 3.1 flow for proposed model*

## 3.2 Literature review

A review of the literature in the field of image processing for each stage of the system was done for this study, including available books, journals, case studies, earlier linked literature, and instructions to comprehend the topic completely.

### 3.3 Problem identification and the objective formulation

The researcher has reviewed different literature and related works previously done related to this thesis work, to identify practical problems in an actual application environment, and the researcher conducted interviews with agricultural experts to know the gaps between existing wheat disease

identification methods. After that, the problem specification is used to infer a solution's goals.The objectives of the study that are inferred from the problem specification are explained.

### 3.4 data collection

Data collection is the most important task when using deep learning. As a result, they require a substantial amount of data for training. In this study, necessary images of wheat healthy leaves, wheat images infected by stem rust disease, images infected by yellow rust disease, and wheat images that are infected by leaf rust disease were collected (372 images from Adet Agricultural Research Centre and 213 images from Arbaminch Agricultural Research Centre in Ethiopia), images collected by plant science experts at different times when disease occurred in wheat for their experimental purposes. And the rest of the 1953 images were collected from publicly available datasets (Kaggle). This helps models benefit from training in various imaging and attribute areas.

Here are some steps we can take to use images taken from different areas together for model training. We combined the datasets and performed preprocessing steps, including image resizing the images to a common size, converting the images to a standardized format to standardize their characteristics such as the color space, brightness, contrast, and sharpness, this can help to reduce the differences between the images and make them more consistent, augmentation, and normalization, to ensure uniformity, compatibility and improve the quality of the data. Using public and local image datasets together for model training can be a useful to improve the accuracy and robustness of wheat rust disease detection and classification models.

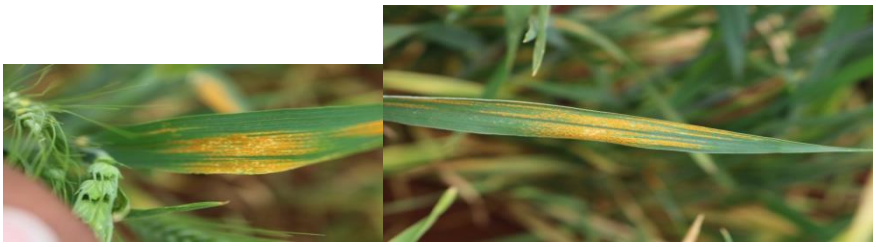


*Figure 3.2 Sample input wheat image before Preprocessing*

## 3.5 Model selection

CNN is an intriguing technique for adaptive image processing. The approach is used to extract features, classify data, train and test models, and assess the model's accuracy. Without the requirement for a separate stage of feature extraction or pre-processing, CNNs use raw data.

The CNN algorithm's key benefit for the identification of wheat rust disease is that it is more reliable and automated than conventional machine learning techniques. In traditional machine learning methods, different algorithms were created for various problems, necessitating the use of more bespoke methods. The following are some of the primary justifications for using CNN in this thesis: Numerous studies published in the past have demonstrated that CNN is the best classification method for computer vision applications, outperforming all others.

- Numerous studies published in the past have demonstrated that CNN is the best classification method for computer vision applications.
- Since CNNs are built to mimic human vision, they perform better on image-related tasks than other deep learning models.
- With the use of pertinent kernels, CNN is able to detect spatial and temporal relationships.

## 3.6 Proposed framework

The proposed architecture for wheat rust disease detection contains the following components: preprocessing, data augmentation, classification, and model evaluation. Preprocessing is the initial element of the suggested model. It performs preprocessing tasks, including image enhancement and normalizing the image to a standard size. In addition to this, the component does the initial task of making the input image ready for feature extraction. The second component of the proposed model is augmentation, which is the task of producing more data from the original data set to increase models performance. After we augment the training data, we developed the proposed

model. The last component of the proposed architecture is the classification and evaluation of the proposed model.

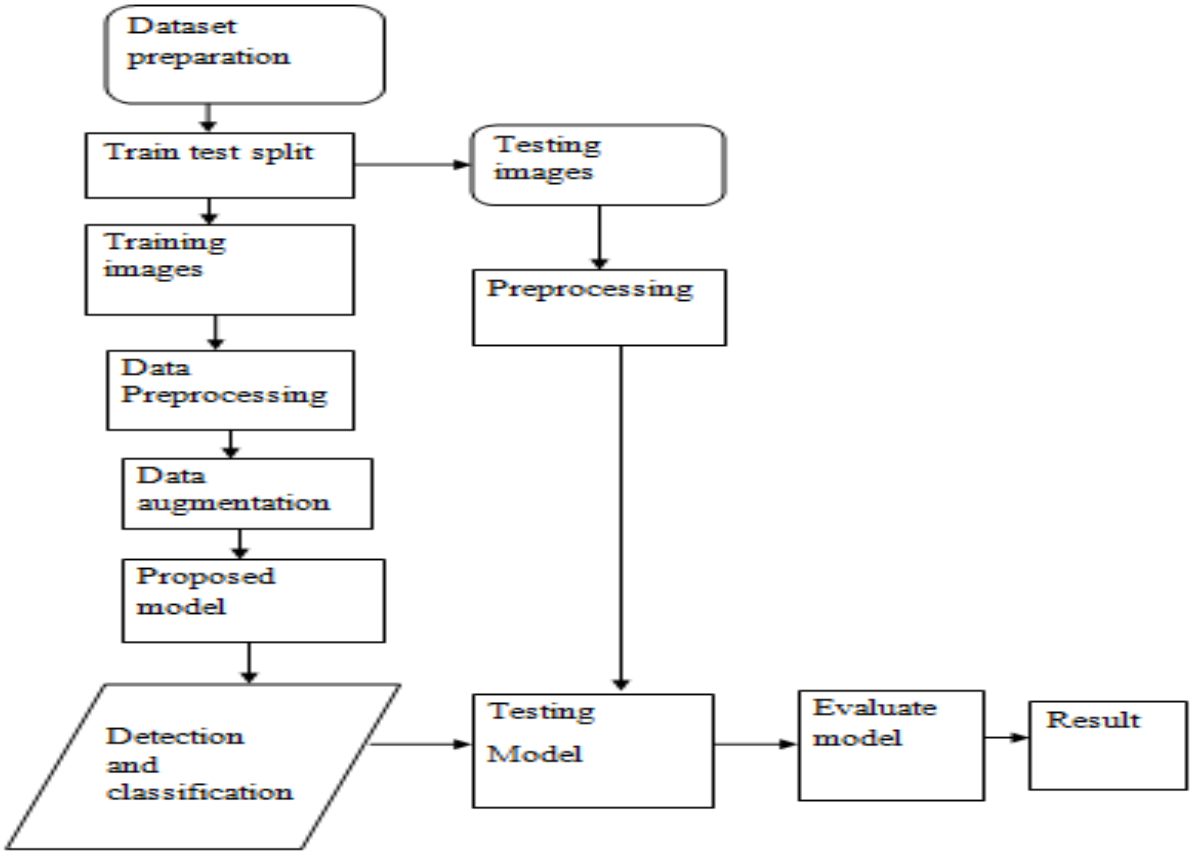



*Figure 3.3 Design of proposed model*

Our proposed model is entirely original. We used Visio software to design the architecture based on our own insights and analysis of the problem. We believe that the specific features and components thatwe have included will make the model effective, and we have provided evidence to support this in our thesis.

### 3.7 materials and tools

#### 3.7.1 Software tools

To select the ideal software package for a deep learning model's implementation in wheat rust disease detection and classification, a review of the software tools that are currently available together with their libraries was done. The investigation revealed that there are tools that are

universal for both machine learning and deep learning methods. Before choosing the tools, we took into account a few factors that can be used to select the appropriate libraries and software tools. The other requirements include selecting tools that have sufficient learning materials, such as free video tutorials and prior knowledge, and the requirement that they be used on machines with limited resources, such as those with only a CPU. We chose Python as a programming language for preprocessing, feature extraction, and classification, with its libraries, Keras and TensorFlow, as robust backend libraries for deep learning, to develop the wheat disease detection model. These tools are made in the well-known programming language Python and satisfy all requirements.

**Anaconda3**–is an open-source and free platform for the Python programming language that is utilized for the model's implementation. Any OS, including Windows, Linux, and Mac OS, can quickly and easily install it. It offers tools that may be used to create deep learning and machine learning models. Python is included along with several IDEs, such as Spyder and Jupyter Notebook, Anaconda Prompt, and others, as part of the Anaconda distribution. As a result, it's a comprehensive solution that's incredibly useful and simple to download and install on your computer. For the coding portion of this thesis, we used Visual Studio and the Jupyter notebook.

**Tensor Flow-**Currently, Tensor Flow is the most widely used open-source, and free deep-learning library. It is a platform for numerical competitions built on the basis of dataflow graphs, and it is compatible with any desktop running Windows, macOS, or Linux, as well as in the cloud as a service and on mobile devices running iOS and Android. It is scalable and significantly more effective for neural networks than other deep learning frameworks and conventional methods of data representation.

Tensor Flow's design is effective for preprocessing data, creating models, refining models, and estimating models. Each calculation in Tensor Flow uses tensors (n-dimensional arrays) to represent different sorts of data. Tensors are used to represent RGB images.

**Keras**is a high-level neural network Python application that offers a simple user interface for building deep learning models that use Tensorflow. Has a very user-friendly interface, is straightforward to extend with Python, and most importantly, contains pre-trained CNN models that we employed in the experiment.

**Visio 2016** is a software tool used to design a diagram or grant chart with the use of ready-made templates, which make difficult information simpler

### 3.7.2 Hardware tools

We have used an Intel(R) Core(TM) i3-2310M CPU @ 2.10GHz with 4GB of memory to write the documentation part of the proposed model and Desktop(DELL E1920HIntel(R) Core(TM) i5-10505 CPU @ 3.20GHz  with 8GB memory)for implementation**.**

## 3.8 Feature extraction

Feature extraction is the process of retrieving meaningful information from an image that is used for the identification of the image in predefined categories [71]. In this study, features are extracted in two phases: The training phase is the initial stage. The testing phase is the next phase, which is responsible for evaluating the performance of the proposed model. Most image classification issues begin with feature extraction, since before the classification step begins; crucial features that are needed to identify the images were extracted using this method. One feature parameter, a color feature, is employed for the detection of the wheat rust disease because, in the conventional method, a person's ability to distinguish between different colors allows them to determine whether or not a crop is afflicted with the disease. As a result, the suggested model establishes the output classes based on the input image's characteristics, which are discovered during training.

.

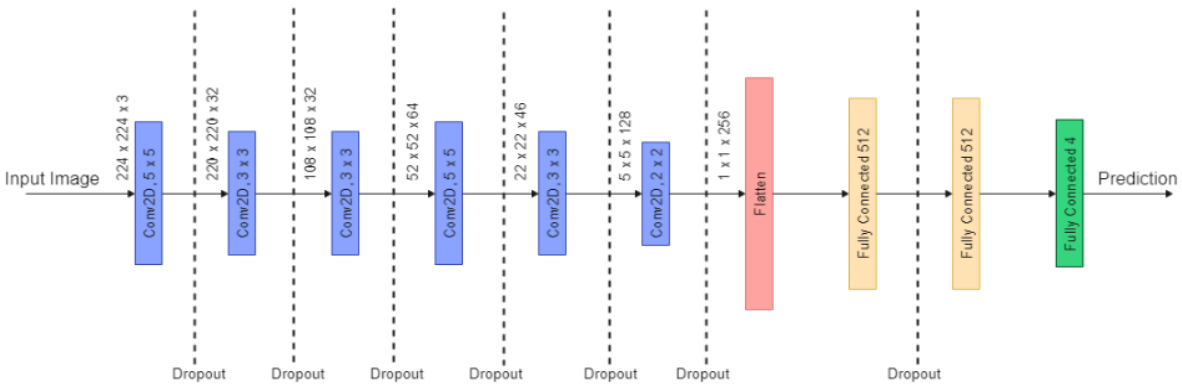



*Figure 3.4 Architecture of feature extractions on wheat rust detection model*

### 3.9 Regularisation method

Regularization is a technique that aids in preventing over fitting by punishing a model for having high weights. In essence, a model has huge weights when it doesn't fit the input data well. Regularization aids in resolving this issue by ensuring that our model fits the data properly, preventing false positives where something appears to be true but is actually untrue.

A collection of techniques called regularization is used in deep learning to lower the generalization error. After training, most models perform exceptionally well on a small subset of the entire data set, but they struggle to generalize. Strategies for regularization seek to minimize training errors while also reducing over fitting.

We refer to this ability to generalize, or the ability of our models to correctly classify data points they weren't trained on, as the capacity of our models, which regularization helps us control. Without regularization, our classifiers may quickly over fit to our training data and become overly complex, which would prevent them from generalizing to our testing data (as well as information from sources other than the testing set, including fresh images from the wild)[72].

### 3.9.1 L1 and L2 Regularization

**L1 and L2 Regularization** approaches reduce the risk of over fitting the training set for two reasons. Some hidden unit weights approach (or reach) 0. As a result, they have less of an impact, and the final network resembles a smaller network, which makes it simpler to grasp. In a simpler network, over fitting is less likely to happen. The input z of a hidden neuron's activation function decreases with decreasing weights. Several activation functions behave linearly for values close to 0. We have used the L1 regularization procedures to reduce some of the model's weights to zero or to a lesser size.

L1 may alternatively represent the total of the absolute weights, while L2 may represent the total of the squared weights. L1/L2 might then be equal to the sum of the squared and absolute weights. The convolution layers, where feature extraction takes place and no learning occurs, are the only places in our network where the L1 and L2 approaches should be used (i.e., where weights are being changed).

L1 and L2 add a new term known as the regularization term and update the general cost function.

Cost function = loss (let's assume binary cross-entropy) + regularization term

The inclusion of this regularization component leads to a decrease in the values of the weight matrices because it is believed that a neural network with smaller weight matrices produces simpler models. As a result, it also significantly lessens over fitting.

Yet the L1 and L2 regularization terms are different.

L2 regularaization we have:

$$Cost\,function = Loss + \frac{\lambda}{2m} * \sum ||w||^2$$

The regularization parameter in this case is lambda. The hyper parameter whose value has been tuned for improved outcomes is this one. Since L2 regularization leads the weights to decay towards zero, it is also known as weight decay (but not exactly zero).

L1 regularaization we have

$$Cost\,function = Loss + \frac{\lambda}{2m} * \sum ||w||$$

We punish the weights' absolute value in this. In contrast to L2, the weights are regressible to zero here. As a result, it is really helpful to condense our model.

The regularizers in Keras allow us to easily apply regularization to any layer.

The samplecode we appled l2regularization to a convulation layer of the ower model:

Model. Add (Conv2D (32,(3,3), kernel_regularizer =l2(0.00001),bias_regularizer=l2(0.00001)

here value 0.00001 is the value of regularaization parameter that is lamda it needs further improvement.

**Dropout**

One of the more intriguing regularization approaches is dropout. It also yields excellent results, making it the most popular regularization method in the deep learning community.

Dropout has the effect of momentarily shrinking the network, and we are aware that less complicated networks are more resistant to overfitting.

Dropout regularization entails the arbitrary removal of some network nodes. Each node has a probability of being disabled during training. Each training example's dropout procedure is carried out individually at random. Hence, a distinct network may be used to train each training sample. Dropout regularization produces a less complex model since it results in a simpler network.

The Keras core layer allows us to implement dropout in Keras. here we have sample source code to implement dropout in convulation layer of our proposed model.

model.add(Conv2D(64,(5,5),kernel_regularizer=l2(0.00001),bias_regularizer=l2(0.00001),activation=’relu’ ,name="conv2d_4"))

model.add (MaxPooling2D (pool_size=(3, 3),name="max_pooling2d_4"))

model.add(Dropout(0.3))

### 3.9.2 Early stopping

Early stopping is a type of cross-validation in which the testing set is kept as a subset of the training set. We immediately halt the model's training when we notice that the testing set's performance is getting worse. This is referred to as an early stop. As the name implies, early stopping entails ending the training phase before the previously specified number of iterations. If we plot the cost function as a function of iterations on both the training set and the validation set, we can see that with an over fitting model, the validation error may start to increase after a predetermined number of iterations, whereas the training error always declines. When the validation error stops decreasing, the training procedure should be stopped. By stopping the training process, we force the model to be simpler and hence reduce over fitting.

Using the callbacks function in Keras, early halting can be applied. A sample of the code we use to achieve the proposed model's early stopping:

Callback=tf.keras.callbacks, Early Stopping (monitor='loss', patience=3)

### 3. 10 Performance evaluation metrics

Accuracy, precision, F1 score and recall were used as evaluation metrics to evaluate how well our model performed.

### 3.10.1 Accuracy

Usually, when we use the word accuracy, we mean categorization accuracy. Out of all the input samples, the percentage of accurate predictions is calculated.

Accuracy is defined as the proportion of correct forecasts to all other predictions.

Accuracy = true positive + true negative / total samples

#### 3.10.3 Precision

To determine precision, the entire number of accurately positive results is divided by the total number of positive results that the classifier anticipated.

Precision = True positive /True positive +False positive

#### 3.10.4 Recall

**Recall**-It is calculated by dividing the total number of pertinent samples by the number of precise positive results (all samples that ought to have been classified as positive).

Recall = true positive/true positive + false negative

### 3.10.5 F1 score

To determine a test's level of accuracy, use the F1 Score. The F1 score is the harmonic mean of memory and precision. The [0, 1] range represents the F1 score. It demonstrates the accuracy and dependability of your classifier (the percentage of cases that were successfully classified; it does not miss a sizable number of examples). Although a very precise categorization is produced by high accuracy but low recall, many examples that are challenging to identify are also missed.

# CHAPTER FOUR

## 4. RESEARCH IMPLEMENTATION

### 4.1 Dataset preparation

For proposed model dataset was prepared by using the image data of wheat that are collected from Adet agricultural research center, Arebamich agricultural research center and additional image data that are taken from publically available datasets. Dataset contains 870 healthy images of wheat and 1686 images infected by yellow rust disease, leaf rust disease and stem rust disease. Dataset which contain total 2538 wheat image was randomly partitioned in to 80% for training and the 20% for testing and also 70% for training and 30% for testing the proposed model.

There are several techniques for splitting a dataset into training and testing sets in deep learning. Here are a few of the most common techniques: Random Split, K-Fold Cross-Validation, Stratified Sampling and etc. Random Split: This technique involves randomly dividing the dataset into two parts, typically with a ratio of 70/30 or 80/20, where one part is used for training the model and the other part is used for testing. We used random split for our dataset, because this technique is simple to implement.

**Training Set Size:** The size of the training set is critical in deep learning because deep learning models are complex and require a large amount of data to learn the patterns in the data. The training set size should be large enough to ensure that the model can learn the patterns in the data effectively. Typically, a training set size of at least 70-80% of the total dataset is recommended.

**Test Set Size**: The test set is used to evaluate the model's performance after training is completed. The test set size should be large enough to provide a reliable estimate of the model's performance but not so large that it reduces the size of the training and validation sets. A test set size of 10-20% of the total dataset is recommended But In our case we used 80/20 and 70/30 split ratio**,** we found that using a 70/30 split ratio provided better accuracy than an 80/20 split ratio. This could be due to a variety of factors, such as the size and complexity of our dataset, the nature of the problem being addressed, and the specific model architecture and hyper parameters being used. It is also possible that the better accuracy on the 70/30 split was due to chance or random variations in the data.

In general, the choice of the dataset split ratio depends on several factors, and it is recommended to use a range of ratios and evaluate the model's performance on each split to determine the optimal ratio.

*Table 4.1 Number and types of images in dataset before split datasets*

| Type of Images | Number of images | For training | For testing |
|---|---|---|---|
| healthy | 870 | 602 | 250 |
| stem rust | 243 | 170 | 73 |
| Leaf rust | 448 | 313 | 135 |
| Yellow rust | 995 | 730 | 265 |

*Table 4.2 Data set split percentage*

| Datasets | Percent | Total number of image |
|---|---|---|
| Training | 70% | 1777 |
| Testing | 30% | 761 |

| Datasets | Percent | Total number of image |
|---|---|---|
| Training | 80% | 2031 |
| Testing | 20% | 507 |

### 4.2 Optimization algorithms

Wheat rust detection model is prepared to update the loads, and the slope plunge advancement calculation is produced to reduce error rates. The slopes are the most well-known and frequently used enhancement calculation in deep learning studies. Additionally, every modern deep learning library includes Keras implementations of angle plummet improvement calculations (used in this postulation). Adam to process versatile learning rates is the most well-known advancement calculation used nowadays for creating deep neural models. Each boundary uses squared angles to scale the learning rate, and it also makes use of the inclination's shifting normal.

### 4.3 Activation function

Wheat rust disease detection model has used three activation functions for use in experiments: In the ReLU (rectified linear activation function) in hidden layers, the softmax and sigmoid activation processes in the model are particularly evident in the output layer.

### 4.4 Epochs

How frequently the entire training set is displayed to the network during training. This is how much forward and backward iteration the whole data set undergoes through the model or network. For this thesis; we used up to 300 epochs to obtain the optimal epoch.

### 4.5 Batch size

It represents the overall amount of training input used during instruction. To update the parameters, the single set is split into smaller subsets known as "mini-batches." However, the model is trained using the entire dataset in order to avoid any loss of accuracy and maintain the broadness of the model.A model for wheat rust disease detection was developed using batch sizes 64 and 32.

### 4.6 Learning rate

The wheat rust disease detection model was trained using back-propagation, and weight changes were based on the learning rate. It regulates how much weight will be updated during back-propagation. Choosing the right learning rate for the experiment was difficult. It takes longer to train the learning rate with a small number than with one that is larger. However, when given a lower value, the model performs better than the one with a higher learning rate. 0.01, 0.001, 0.0001, 0.00001, and 0.000001 learning rates were used in the experiment. This was done in order to select the ideal learning rate.

### 4.7 Image preprocessing

Image processing is a procedure that aims to improve the quality of images. The preprocessing task is the basic step in an image processing application that can help get a more meaningful interpretation of an image. The preprocessing stage creates an improved image and output from a raw image as an input. Additionally, it may have an effect on a classifier model's general performance. Various preprocessing procedures, including image resizing and augmentation, are used in this study. Image resizing is the first image preprocessing task that focuses on preserving

an important region of an image, minimizing distortion, and improving efficiency. As the size of an image increases, it requires more computer resources to process; hence, there is no single standard for how to resize images; instead, we resized depending on the computational resources we have. All imported images in this study have a constant size of 224*224 because using images of different sizes can alter how accurately features are extracted. And then images are converted into numpy arrays that Keras can work with.

```
In [15]: input_shape=(224,224,3)
train_generator =train_datagen.flow_from_directory(train_dir,target_size=(224,224),batch_size=64)
test_generator=test_datagen.flow_from_directory(test_dir,shuffle=True,target_size=(224,224),batch_size=64)

Found 1815 images belonging to 4 classes.
Found 723 images belonging to 4 classes.
```

Source code to resize image data

*Figure 4.1Sample resized images*

```
In [2]: def get_files(directory):
          if not os.path.exists(directory):
            return 0
          count=0
          # crawls inside folders
          for current_path,dirs,files in os.walk(directory):
            for dr in dirs:
              count+= len(glob.glob(os.path.join(current_path,dr+"/*")))
          return count
        train_dir ="C:/Users/3090/Desktop/anchinesh/train"
        test_dir= "C:/Users/3090/Desktop/anchinesh/test"
```

```
In [3]: train_samples =get_files(train_dir)
        #to get tags
        num_classes=len(glob.glob(train_dir+"/*"))
        #test file image count
        test_samples=get_files(test_dir)
        print(num_classes,"Classes")
        print(train_samples,"train images")
        print(test_samples,"test images")
```

*Source code to read and convert images into numpy array for proposed model*

```
In [22]: train_generator[0]
Out[22]: (array([[[[0.        , 0.        , 0.        ],
                   [0.        , 0.        , 0.        ],
                   [0.        , 0.        , 0.        ],
                   ...,
                   [0.        , 0.        , 0.        ],
                   [0.        , 0.        , 0.        ],
                   [0.        , 0.        , 0.        ]],

                  [[0.        , 0.        , 0.        ],
                   [0.        , 0.        , 0.        ],
                   [0.        , 0.        , 0.        ],
                   ...,
                   [0.        , 0.        , 0.        ],
                   [0.        , 0.        , 0.        ],
                   [0.        , 0.        , 0.        ]],
```

*After converted into numpy array*

### 4.8 Image augmentation

It assists the network in extracting more intricate characteristics from the data and mitigates the over fitting issue[73]. Different data augmentation approaches have been used for this study on the initial images that were collected. To accomplish this, the Keras deep learning library was used to transform, random shift, random rotate, randomly scale, random zoom, random erasing, horizontal shift, vertical shift, and fit models via the image. To experiment with the data augmentation technique, a Python package was used. Augmentation is done live during training. What happens is that the augmentations are applied every time a sample is loaded. In the case of

the random resize crop, each sample is cropped randomly and then resized for each epoch. This means that the model does not see the exact same image each epoch, which improves generalization and prevents over fitting**.**

The size of an image dataset after augmentation depends on several factors, such as the size of the original dataset, the specific implementation of the data augmentation technique, and the size of each transformed image. However, we can estimate the size of the augmented dataset based on the number of transformations applied and the size of the original dataset.

In our case the original dataset contains 2553 images, and the transformation type is 5 as we seen below screen shut source code, which means that 5 different transformations are applied to each image, the total number of new images in the augmented dataset will be:

Total number of new images = Number of transformations x Size of original dataset

Total number of new images = 5 x 2553 = 12,765

We have total of 15318 images after augmentation.

Algorithms applied for image augmentation:

In put: Preprocessed image dataset

Start

- Rescale ,
- Initialize rotation range,
- Width shift range,
- Height shift range,
- Shear range,
- Zoom range,
- Horizontal flip,
- Vertical flip

Return Aug: //augmented images

End

During augmentation, images are not stored on a disc and don't require storage memory; rather, the transformed images are generated at run-time.

```
In [4]: train_datagen=ImageDataGenerator(
            rescale=1./255,
            shear_range=0.2,
            zoom_range=0.2,
            rotation_range=40,
            width_shift_range=0.2,
            height_shift_range=0.2,
            horizontal_flip=True,
        )
        test_datagen=ImageDataGenerator(rescale=1./255)
```

Source code for data augmentation

## 4.9 Over view of wheat rust disease detection model

**Input layer:**The model's first convolutional layer uses 32 kernels with RGB images of sizes 224*224*3 with four different classifications( wheat leaf rust, wheat yellow rust, wheat stem rust, and healthy wheat images) that are accepted by the input layer of our wheat rust disease detection model. This layer does not do any calculations; it just delivers the input to the first convolution layer, and this layer has no parameters.

**Convolution Layer**

It serves as the foundational layer of the CNN architecture. The drawn characters from the input image are removed using this method. For character extraction, the input image is used. In order to maintain the association between pixels, the convolution layer continuously learns from the

input data's image features. The mathematical operation requires a 3x3 kernel filter and two input image matrices. The convolution filter of 3 x 3 and the 5 x 5 image pixel value in order to create a map and place the filter value there, CNN first adds the result and divides by the total number of pixels. This layer sets the filter on the image in each available location. Here is an example of CNN utilizing Conv2D, and a few layers from the wheat rust disease detection model.

There are six convolution layers in the wheat rust detection model. The model's first convolution layer applies 32 kernels with a 5 x 5 x 3 pixel size and a 2 pixel stride to filter the 224*224*3 input image, which has 2432 total parameters.

The output of the first convolution layer is sent to the next layer, which filters it using 32 kernels that are 3 by 3 by 32 in size with 9248 parameters. Each of the third and fourth convolution layers has 64 kernels, with 64 kernels of size 3 by 3 by 64 having 18496 parameters and 64 kernels of size 5 by 5 by 64 having 102464 parameters, respectively. The fifth and sixth convolution layers also each include 128 kernels of size 3 by 3 with 73856 parameters and 256 kernels of size 1 by 1 with 33024 parameters, respectively. The ReLuactivation functions are applied across all convolution layers of the wheat rust disease detection model.

**Max_pooling layer**

MaxPooling2D's combination of the use of a moving window across 2D input space with a convolutional layer, Bach normalization, and Before training, hyper parameters are chosen; they stand for the factors that govern the structure in terms of the neural network and the training procedure.

The wheat rust disease detection model has six maximum-poling layers. The first max pooling layer uses a filter size of 3*3 with stride 2 to lower the output of the first convolutional layer. In the rest max pooling layer, which pools data using the 2*2 filters of stride 2 as in max pooling, the output of all other convolutional layer is delivered.

**Fully connected layer-** There are 512 input layers and 131584 parameters in the first fully connected layer. The second fully connected layers receive from the first dense layer and produce the classes output into 4 classes by using softmax activation function. The softmax activation function produces an image that is classified as: Dense (4, softmax =activation) Print the model's executive summary. We have 635,812 total parameters in the model.

The 635,812 parameters in our model represent the learnable weights and biases that are used to make predictions. These parameters were determined during the training process, where the model learns to adjust these values to minimize the loss function and improve performance on the task at hand.

```
In [9]: model = Sequential()
        model.add(Conv2D(32, (5, 5),input_shape=input_shape,activation='relu',name="conv2d_1"))
        model.add(MaxPooling2D(pool_size=(3, 3),name="max_pooling2d_1"))

        model.add(Conv2D(32, (3, 3),kernel_regularizer=l2(0.0001), bias_regularizer=l2(0.0001),activation='relu',name="conv2d_2"))
        model.add(MaxPooling2D(pool_size=(2, 2),name="max_pooling2d_2"))

        model.add(Conv2D(64, (3, 3),kernel_regularizer=l2(0.0001), bias_regularizer=l2(0.0001),activation='relu',name="conv2d_3"))
        model.add(MaxPooling2D(pool_size=(2, 2),name="max_pooling2d_3"))

        model.add(Conv2D(64, (5, 5),kernel_regularizer=l2(0.0001), bias_regularizer=l2(0.0001),activation='relu',name="conv2d_4"))
        model.add(MaxPooling2D(pool_size=(2, 2),name="max_pooling2d_4"))

        model.add(Conv2D(128, (3, 3),kernel_regularizer=l2(0.0001), bias_regularizer=l2(0.0001),activation='relu',name="conv2d_5"))
        model.add(MaxPooling2D(pool_size=(2, 2),name="max_pooling2d_5"))
        model.add(Dropout(0.25))
        model.add(Conv2D(256, (1, 1),kernel_regularizer=l2(0.0001), bias_regularizer=l2(0.0001),activation='relu',name="conv2d_6"))
        model.add(MaxPooling2D(pool_size=(2, 2),name="max_pooling2d_6"))
        model.add(Dropout(0.25))
        model.add(Flatten(name="flatten_1"))
        model.add(Dense(512,activation='relu',kernel_regularizer=l2(0.0001),
                    bias_regularizer=l2(0.001)))
        model.add(Dropout(0.25))

        model.add(Dense(512,activation='relu'))
        model.add(Dense(4, activation='softmax',kernel_regularizer=l2(0.0001),
                    bias_regularizer=l2(0.0001)))
        callback = tf.keras.callbacks.EarlyStopping(monitor='loss', patience=5)

        model.summary()
```

*Figure 4.2 layers, hyper parameters, and regularization algorithms for proposed model*

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_1 (Conv2D)           (None, 220, 220, 32)      2432

 max_pooling2d_1 (MaxPooling  (None, 73, 73, 32)       0
 2D)

 conv2d_2 (Conv2D)           (None, 71, 71, 32)        9248

 max_pooling2d_2 (MaxPooling  (None, 35, 35, 32)       0
 2D)

 conv2d_3 (Conv2D)           (None, 33, 33, 64)        18496

 max_pooling2d_3 (MaxPooling  (None, 16, 16, 64)       0
 2D)

 conv2d_4 (Conv2D)           (None, 12, 12, 64)        102464

 max_pooling2d_4 (MaxPooling  (None, 6, 6, 64)         0
 2D)

 conv2d_5 (Conv2D)           (None, 4, 4, 128)         73856

 max_pooling2d_5 (MaxPooling  (None, 2, 2, 128)        0
 2D)

 dropout (Dropout)           (None, 2, 2, 128)         0

 conv2d_6 (Conv2D)           (None, 2, 2, 256)         33024

 max_pooling2d_6 (MaxPooling  (None, 1, 1, 256)        0
 2D)

 dropout_1 (Dropout)         (None, 1, 1, 256)         0

 flatten_1 (Flatten)         (None, 256)               0

 dense (Dense)               (None, 512)               131584

 dropout_2 (Dropout)         (None, 512)               0

 dense_1 (Dense)             (None, 512)               262656

 dense_2 (Dense)             (None, 4)                 2052

=================================================================
Total params: 635,812
Trainable params: 635,812
Non-trainable params: 0
_________________________________________________________________
```

*Figure 4.2 Summary of wheat rust disease detection model*

### 4.9.1 Detection of wheat rust disease using pre-trained models

Pre-trained models can provide a good starting point for developing new models and can significantly reduce the time and resources required for training. In our case, we sounds like to use a pre-trained model to evaluate the accuracy of our dataset and to develop a new proposed model with lower parameters that is compatible with the resources available to farmers.

When creating deep learning models from scratch, a large amount of training data is needed for the network parameters to converge on the best answer that generalizes the problem at hand. On the other side, the network has a tendency to over fit the supplied cases with a smaller dataset.
The use of the transfer learning approach can mostly prevent such obstacles.
The idea of transfer learning is to reuse the network that has already been trained and whose weights (parameters) have grown enough to converge on a large dataset. Considering the issue at hand, a transfer learning strategy can be chosen.
Many images are used to train some CNN models, generally on a massive image classification assignment with countless classes. The model's capacity to generalize an object is higher due to the training data used, which consists of millions of images and thousands of classes. As a result, even though the challenges are distinct from those of the initial work,
Transfer learning is the method used to train and use these models, which are referred to as "pre-trained models." It can be difficult or even computationally expensive to train these designs from scratch. The knowledge base, also known as the convolutional base component, is made up of the characteristics that were learned during training. Convolution layers close to the input layer in the convolution base discover features common to all the given images.
In transfer learning, there are two methods that are frequently employed: The first is to modify only the fully linked layer while training the complete convolutional basis of the learned model. The other method is to train a portion of the convolution base while the trained model's weights are frozen. Fine-tuning is the process of training certain convolutional basis components. Using our dataset, we trained two pre-trained models, vgg19 and Resnet50, and compared the outcomes with the suggested model.

## The experiment

This thesis takes three scenarios into account. In the first two instances, the wheat rust disease is detected and classified using a pre-trained method, and in the third scenario, a new deep learning-based architecture is suggested. The two well-known CNN architectures—VGG-19 and Resnet50—that triumphed in the biggest image classification competition were picked during transfer learning. The proposed model has only 635,812 parameters, the result of millions of parameters being reduced in the CNN model. The 635,812 parameters in my proposed model refer to the learnable weights and biases that the model uses to make predictions. This number is relatively small because I used techniques such as convolutional neural networks to reduce the

dimensionality of the data and extract relevant features. By reducing the number of parameters, I aimed to create a more efficient and lightweight model that can make accurate predictions while minimizing computational resources."

## 4.11 Model Implementation, Evaluation of result and Discussion

The experiment tests the detection performance using three different classification settings. The first two cases use pre-trained CNN models, while the third case employs the suggested model. Our studies consist of two primary phases, similar to the majority of deep learning classification models. The training phase is the first, and the testing phase is the second. Throughout the training phase, the classifier is repeatedly shown data, and weights are changed to get the intended outcome. To assess the effectiveness of the detection method, the classifier applies the training algorithm to data that has never been seen before (test data). The details of the experimental findings are provided below.

### 4.9.2 Detection of wheat rust disease Using VGG19 pre-trained model

There are only 33 convolution layers in the VGG model, which are anchored on one another in increasing layer depth; it is simple. The VGG model comes in two different iterations: the first is the VGG16, which has 16 weight layers, and the next is the VGG19, which has 19 weight layers. It takes 224*224 RGB images as input and outputs 1000 classes from the 14 million-images of Image Net dataset. The input is routed via the model's stacked convolution layers with a 3-by-3 receptive field before being subjected to ReLU's nonlinearity. The number of classes found using a Softmax activation function in the Image Net dataset is equal to the channel depth of the first two layers (4096) and the final layer (1000).

A down-sampled RGB image with a size of 224 by 224 is used as the model's input in our experiment, and the model is then modified to produce two classes of output in our dataset. A staggering 138 million parameters make up the original VGG-19 model. Because the spatial dimension of the image in our model is smaller and we only trained some of the model, we trained the model using 20,024,384 parameters. In order to train the model, the complete network convolution base is used, and only the fully connected layers are changed. The results show a considerable amount of over fitting because of the model weights being learned using countless images that are not part of our dataset, when we try to train the model with only 2538 original images, over fitting happens because there are innumerable images that are not in our dataset and thousands of classes.As a result, we have chosen to freeze portions of the model's layers (or "conv

blocks") in order to run a separate experiment with the supplemented data. The following are the hyper parameters we have used during network training for vgg19.

*Table 4.3 hyperparameters used to train vgg19*

| Parameters | Value |
|---|---|
| Epoch | 300 |
| Batch size | 32 |
| Optimization algorithms | SoftMax |
| Learning rate | 0.00001 |

```
[1]   1 from keras.layers import Input, Lambda, Dense, Flatten
      2 from keras.models import Model, load_model
      3 from keras.applications.vgg19 import VGG19
      4 from keras.applications.vgg19 import preprocess_input
      5 from keras.preprocessing import image
      6 from keras.preprocessing.image import ImageDataGenerator
      7 from keras.models import Sequential
      8 import numpy as np
      9 from glob import glob
     10 import matplotlib.pyplot as plt
```

Source code to import vGG19 and necessary packages to train vgg19 with our dataset

```
3 # RE-SIZE ALL THE IMAGES TO THIS
4 IMAGE_SIZE = [224,224]
5 train_path= r'/content/drive/MyDrive/anchinesh/train'
6 valid_path= r'/content/drive/MyDrive/anchineh/test'
```

```
[ ]   1 # IMPORTING THE VGG 19 LIBRARY AS SHOWN AND ADDING PRE PROCESSING LAYER TO THE FRONT OF VGG
      2 # WE WILL USE IMAGENET WEIGHTS
      3
      4 vgg19= VGG19(input_shape= IMAGE_SIZE + [3], weights='imagenet',include_top=False)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg19/vgg19_weights_tf_dim_order
80134624/80134624 [==============================] - 0s 0us/step
```

```
[ ]  1 # don't train existing weights
     2 for layer in vgg19.layers:
     3     layer.trainable = False
```

```
[ ]  1
     2 # useful for getting number of output classes, here we have 4 class
```

```
[ ]  1
```

```
[ ]  1 folders = glob('/content/drive/MyDrive/anchinesh/train/*')
```

```
# don't train existing weights
for layer in vgg16.layers:
    layer.trainable = False
```

```

# useful for getting number of output classes, here we have 4 class
```

```

```

```
folders = glob('/content/drive/MyDrive/Geschlossen/Breite/*')
```

```

```

```

folders
```

```
['/content/drive/MyDrive/Geschlossen/Breite/Geoff_mask',
 '/content/drive/MyDrive/Geschlossen/Breite/Klee_mask',
 '/content/drive/MyDrive/Geschlossen/Breite/Seurat',
 '/content/drive/MyDrive/Geschlossen/Breite/getting_mask']
```

```

# FLATTENING THE LAST LAYER OF VGG16
x= Flatten()(vgg16.output)
```

```
# we will be having 4 nodes, so our output layer because we have 4 classes, to classify
prediction= Dense(len(folders),activation="softmax")(x)

# creating a model
model= Model(inputs=vgg16.input,outputs=prediction)
```

```

# viewing the structure of model
# in last layer we can see that we have  4 nodes.
model.summary()
```

```
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 224, 224, 3)]     0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 224, 224, 64)      1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 224, 224, 64)      36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 112, 112, 64)      0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 112, 112, 128)     73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 112, 112, 128)     147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 56, 56, 128)       0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 56, 56, 256)       295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_conv4 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 28, 28, 256)       0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 28, 28, 512)       1180160
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_conv4 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 14, 14, 512)       0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv4 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 7, 7, 512)         0
_________________________________________________________________
flatten (Flatten)            (None, 25088)             0
_________________________________________________________________
dense (Dense)                (None, 4)                 100356
=================================================================
Total params: 20,124,356
Trainable params: 100,356
Non-trainable params: 20,024,384
_________________________________________________________________
```

```
[ ]  1 model.compile(
     2   loss='categorical_crossentropy',
     3   optimizer='adam',
     4   metrics=['accuracy']
     5 )
```

```
[ ]  1 from keras.preprocessing.image import ImageDataGenerator
```

```
[ ]  1
     2 train_datagen = ImageDataGenerator(rescale = 1./255,
     3                                    shear_range = 0.2,
     4                                    zoom_range = 0.2,
     5                                    rotation_range=20,
     6                                    width_shift_range=0.2,
     7                                    height_shift_range=0.2,
     8                                    horizontal_flip = True)
     9 test_datagen=ImageDataGenerator(rescale=1./255)
    10 test_datagen = ImageDataGenerator(rescale = 1./255)
    11 # Make sure you provide the same target size as initialied for the image size
    12 training_set = train_datagen.flow_from_directory('/content/drive/MyDrive/anchinesh/train',
    13                                                  target_size = (224, 224),
    14                                                  batch_size = 64,
    15                                                  class_mode = 'categorical')
    16
    17
    18
    19
    20 test_set = test_datagen.flow_from_directory('/content/drive/MyDrive/anchinesh/test',
    21                                             target_size = (224, 224),
    22                                             batch_size = 64,
    23                                             class_mode = 'categorical')
```

Found 1619 images belonging to 4 classes.

```
1 # fiting the model
2
3 r = model.fit(
4   training_set,
5   validation_data=test_set,
6   epochs=200,
7   steps_per_epoch=len(training_set),
8   validation_steps=len(test_set)
```

The above screen shout images code shows all sources to train vgg19 with our dataset

#### 4.9.2.1 Result analaysis of vgg19 model by using our dataset

According to the study's findings, the mean test accuracy was 81.7%, and the mean training accuracy was 88%. Using classification accuracy metrics like training and validation accuracy, training loss, and validation loss, the following two charts show the detection accuracy and loss of the VGG19 pre-trained model with respect to epochs. We conducted an experiment by making some modifications to the pre-trained models original design in order to improve the models classification performance in our dataset. The training accuracy is approximately 74% in the first epoch, increasing slightly to pass 88% at epoch 300. The following plots show that, in general, the testing accuracy line and the training accuracy line are nearly parallel and that the testing loss line and the training loss are similarly parallel.

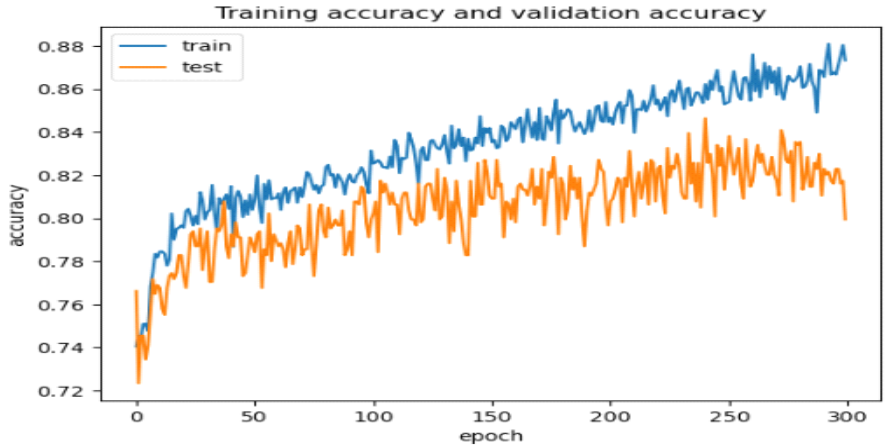



*Figure 4.3training and testing accuracy of vgg19 model*

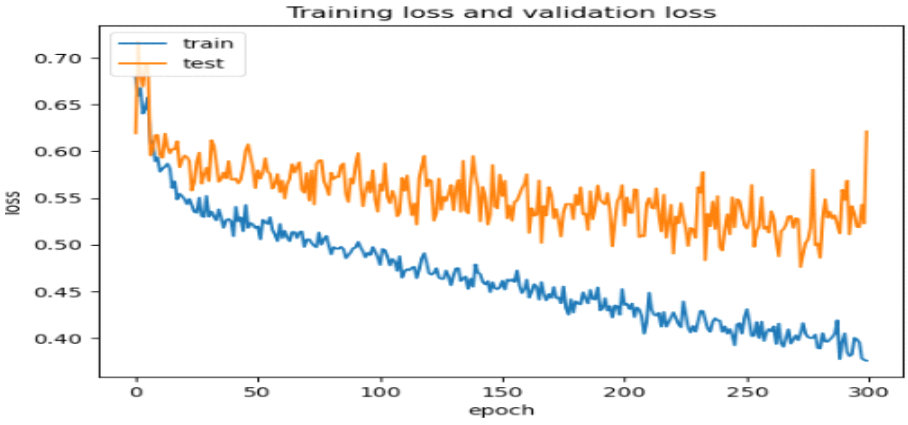



*Figure 4.4 traning and testing accuracy of vgg19 model*

```
In [29]: print('Testing the model')
         final_result = model.evaluate(
             test_data_datagen,
             steps = 723)

         Testing the model
         723/723 [==============================] - 37s 51ms/step - loss: 0.5157 - accuracy: 0.8174
```

Testing accuracy of vgg19 with source code

### 4.9.3 Detection of wheat rust disease Using ResNet-50 pre-trained model with our dataset

A trained deep learning model for image classification called ResNet-50 is available for convolution neural networks (CNN, or ConvNet), a class of deep neural networks most frequently used to analyses view of vision. The millions of images in the Image Net database were used to train ResNet-50, and the weights have been preserved. While using the transfer learning method, these weights can be applied.

Additionally, the model has approximately 23 million trainable parameters, demonstrating a sophisticated design that enhances image recognition. ResNet is an architecture made up of residual neural networks. Identity links help to distinguish a residual network. Each residual block's end is reached immediately by identity connections, which take the input. Each residual block consists of three layers with 1*1 and 3*3 convolutions. In typical neural networks, each layer feeds into the one after it. In a network with residual blocks, each layer feeds into the one below it and then directly into the levels that are around 2-3 hops away. Identity links are the names given to these relationships. Feature extraction and fine tuning are the two applications of transfer learning. The fine-tuning strategy is the main focus of this solution to the design problem. A fully connected layer is built on top of the pre-trained ResNet-50 base model in the fine-tuning method. The weights are not updated, and the learning process is stopped since the complete pre-trained model is frozen.

After training the fully connected layer added to the ResNet-50 base for a limited number of epochs, the 5c block of the pre-trained ResNet-50 model is made trainable by unfreezing the pertinent layers. Unfrozen layers are trained and inserted, together with a fully linked layer, in order to train the data and produce a classification result. A convolution neural network's early layers teach it low-level features, and as it learns deeper layers, it teaches it higher-level features. Hence, the upper layers are defrosted and utilized in the training process. The ResNet-50 model's required dimension is (224, 224, 3). To accommodate this, all images are transformed or resized

before being fed into the pre-trained model. To categories the images, we have adjusted the pre-trained ResNet-50 model and added a fully connected layer on top.

*Table 4.4 hyperparametrs for pretraind Resnet_50*

| Parameters | Value |
|---|---|
| epoch | 200 |
| Batch size | 64 |
| Optimization algorithms | SoftMax |
| Learning rate | 0.001 |

```
[ ]

    from tensorflow.keras.applications.resnet50 import ResNet50
    from keras.layers import Input, Lambda, Dense, Flatten
    from keras.models import Model, load_model
    from keras.preprocessing import image
    from keras.preprocessing.image import ImageDataGenerator
    from keras.models import Sequential
    import numpy as np
    from glob import glob
    import matplotlib.pyplot as plt
```

Source code to import resnet50 and other necessary packages

```
[ ]
    # re-size all the images
    IMAGE_SIZE = [224, 224]
    train_path = '/content/drive/MyDrive/anchinesh/train'
    valid_path = '/content/drive/MyDrive/anchinesh/test'
```

Source code to resize images

```
[ ]
    resnet = ResNet50(input_shape=IMAGE_SIZE + [3], weights='imagenet', include_top=False)

    Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/resnet/resnet50_weights_tf_dim_o
    94765736/94765736 [==============================] - 1s 0us/step
```

Source code to downloading resnet50

```
In [12]: model = models.Sequential()
         model.add(Resnet50)
         model.add(layers.Flatten())
         model.add(layers.Dropout(0.5))
         model.add(layers.Dense(2048, activation='relu'))
         model.add(layers.Dense(4, activation = 'softmax'))
         model.summary()
```

Source code to adding fully connected layer

Final CNN model with fully connected layer and the ResNet50 base after unfreezing the ResNet50 block

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 resnet50 (Functional)       (None, 7, 7, 2048)        23587712

 flatten (Flatten)           (None, 100352)            0

 dropout (Dropout)           (None, 100352)            0

 dense (Dense)               (None, 2048)              205522944

 dense_1 (Dense)             (None, 4)                 8196

=================================================================
Total params: 229,118,852
Trainable params: 209,996,804
Non-trainable params: 19,122,048
```

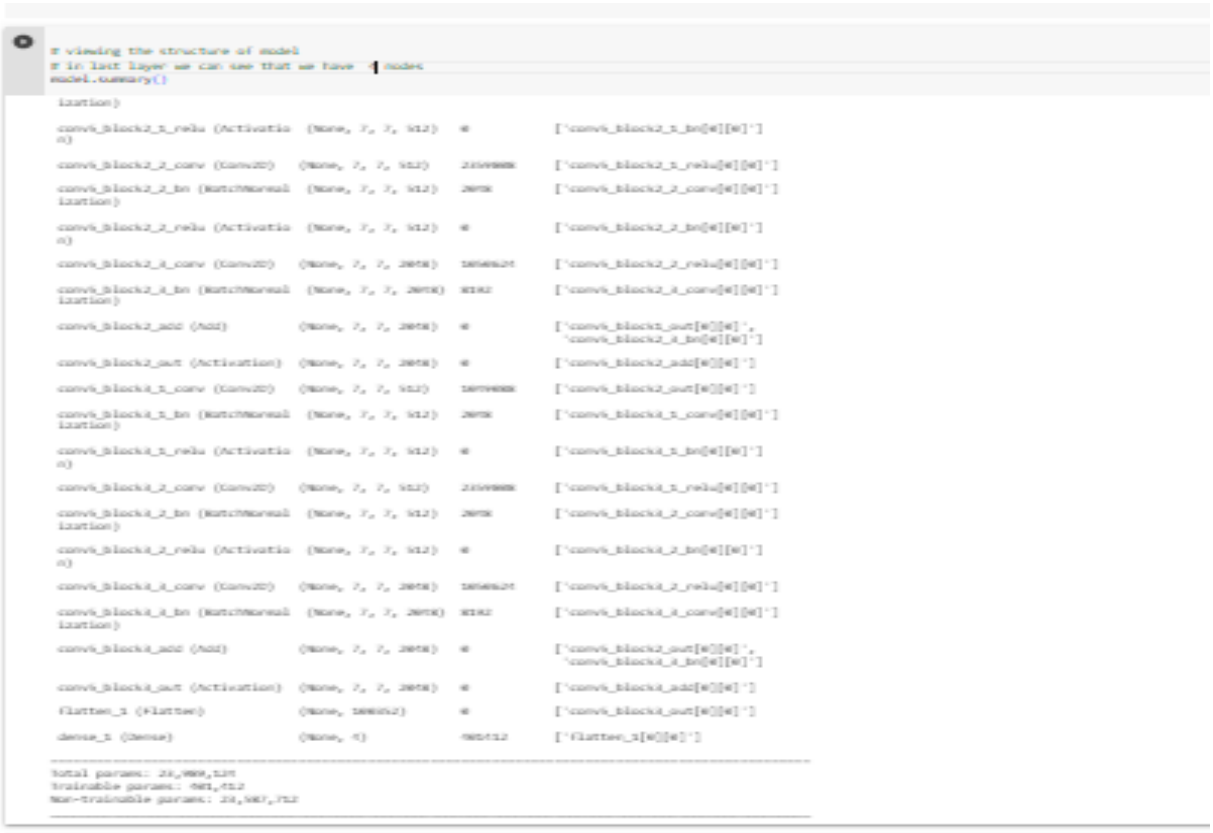
```
In [5]: # don't train existing weights
        for layer in resnet.layers:
            layer.trainable = False
```

Source code to freeze pertained ResNet-50

```
[ ]
    # useful for getting number of output classes, here we have 4 class
    folders= glob(r'/content/drive/MyDrive/anchinesh/train*')
```

```
[ ]
```

```
[ ]
    # useful for getting number of output classes, here we have 4 Class
    folders= glob(r'/content/drive/MyDrive/wanchinesh/train*')
```



```
[ ]  model.compile(
       loss="categorical_crossentropy",
       optimizer='adam',
       metrics=['accuracy']
     )
```

```
[ ]   # Use the Image Data Generator to import the images from the dataset
     from keras.preprocessing.image import ImageDataGenerator

     train_datagen = ImageDataGenerator(rescale = 1./255,
                                        shear_range = 0.2,
                                        zoom_range = 0.2,
                                        horizontal_flip = True)

     test_datagen = ImageDataGenerator(rescale = 1./255)
```

```
[ ]  # providing the same target size as initialied for the image size
     training_set = train_datagen.flow_from_directory(r'/content/drive/MyDrive/a
                                                      target_size = (224, 224),
                                                      batch_size = 32,
                                                      class_mode = 'categorical'



     test_set = test_datagen.flow_from_directory(r'/content/drive/MyDrive/anchle
                                                 target_size = (224, 224),
                                                 batch_size = 32,
                                                 class_mode = 'categorical')
```

```
Found 1618 images belonging to 4 classes.
Found 403 images belonging to 4 classes.
```

```
# fiting the model

r = model.fit(
  training_set,
  validation_data=test_set,
  epochs=5, #10,15,20
  steps_per_epoch=len(training_set),
  validation_steps=len(test_set)
```

```
[ ]  model.save("my_wheat_model.h5")
```

```
[ ]
```

```
[ ]
     # plot the loss
     plt.plot(r.history['loss'], label='train loss')
     plt.plot(r.history['val_loss'], label='val loss')
     plt.legend()
     plt.show()
     plt.savefig('LossVal_loss')

     # plot the accuracy
     plt.plot(r.history['accuracy'], label='train acc')
     plt.plot(r.history['val_accuracy'], label='val acc')
     plt.legend()
     plt.show()
     plt.savefig('AccVal_acc')
```

All source codes we used to train Resnet50 with our dataset

### 4.9.3.1 Result analaysis of the wheat rust detection using pretraind Resnet_50 with our dataset

The following two plots show the detection accuracy and loss of the Resnet_50 pre-trained model with respect to epochs. The experimental results indicate a mean test accuracy of 84.8% and a mean training accuracy of 96%.We conducted an experiment by making some modifications to the pre-trained models original design in order to improve the models classification performance in our dataset. The training accuracy is approximately 89% in the first epoch, increasing slightly to pass 96% at epoch 300.

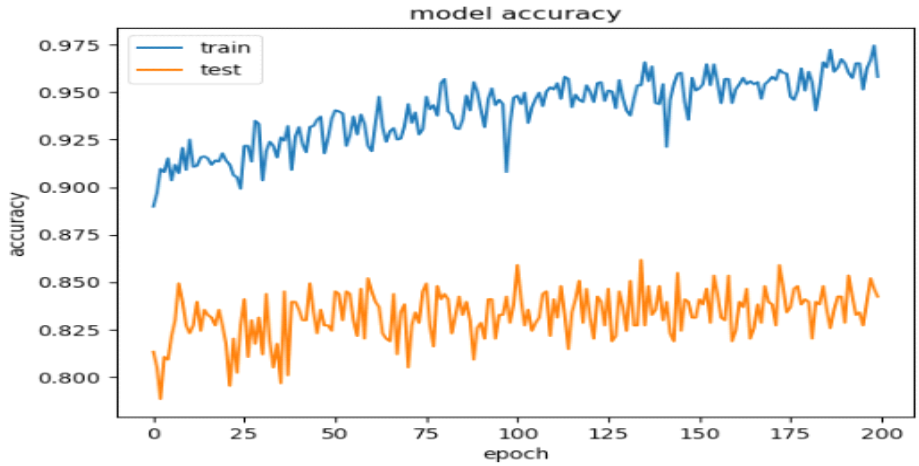



*Figure 4.5 trainingand testing accuracyforResNet_50 model*

```
In [22]: print('Testing the model')
         final_result = model.evaluate(
             test_data_datagen,
             steps = 723)

         Testing the model
         723/723 [==============================] - 42s 57ms/step - loss: 0.6850 - accuracy: 0.8479
```

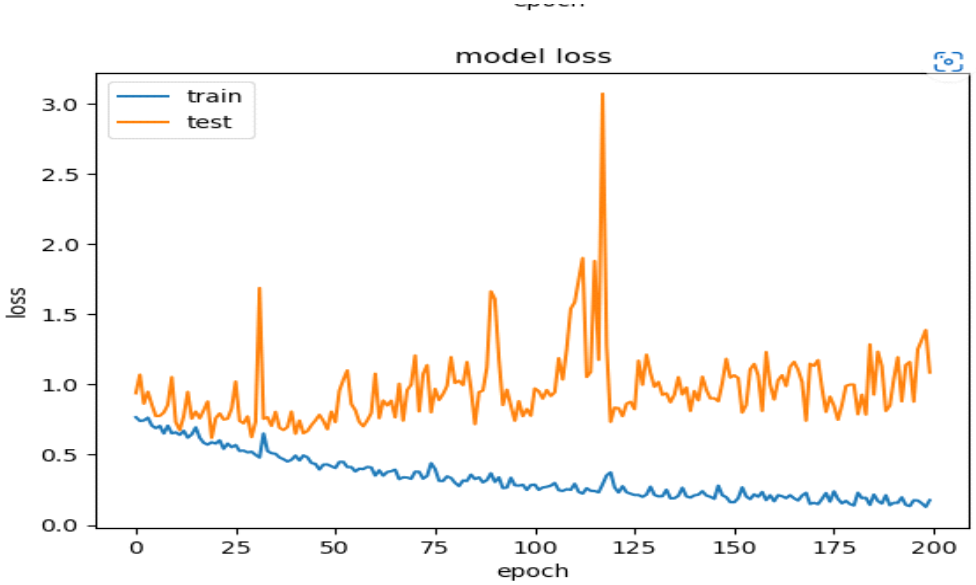



*Figure 4.6 training and testing loss for ResNet_50 model*

### 4.10 Detection of wheat rust disease using proposed model

The proposed model receives 224 x 224 images of any size and outputs four classes. Following the use of data augmentation during the training of the dataset, the suggested model is trained with a total of 2538 wheat images. The suggested model performs numerous experiments by varying the ratio of the training and testing datasets, employing various learning rates, and ultimately employing various activation functions.

## .4.10.1 Scenario one _Before regularization technique

**Result analysis of the developed model by employing different train and test split ratios**

The following table uses the model detection accuracy measurements in the form of percentages for the training dataset and test dataset separately to show the results of the proposed model's experiment employing a different training/testing split ratio.

*Table 4.5 training and testing accuracy and loss for proposed model using different dataset split*

| training split | test split | training accuracy | testing accuracy | training loss | testing loss |
|---|---|---|---|---|---|
| 70% | 30% | 95.5% | 86.5% | 1.4% | 5.2% |
| 80% | 20% | 94% | 86.3% | 1.6% | 5.7% |

The results from the experiments using different ratios of training and testing datasets for the proposed model are encouraging, as shown in Table 4.5.In comparison to the experiment using 80% for training and 20% for testing, the experiment using 70% for training and 30% for testing had better results. A ratio of 7:3 suggests that 70% of the entire dataset is utilized in training and 30% is utilized for testing.

*Table 4.6 Training and testing accuracy and loss for proposed model using differentlearningrate*

| Learning rate | Training accuracy | Testing accuracy | Training loss | Testing loss |
|---|---|---|---|---|
| 0.000001 | 95.5% | 86.5% | 1.4% | 5.2% |
| 0.00001 | 94.5% | 86.3% | 1.6% | 5.7% |
| 0.0001 | 94% | 86.2% | 1.7% | 5.8% |
| 0.001 | 90.4% | 84% | 2.8% | 4.7% |
| 0.01 | 87% | 84.5% | 3.15% | 4.72% |

As shown in table 4.6 above, as the learning rate decreases from 0.01 to 0.000001, the testing and training accuracy of the proposed model increase from 84.5% to 86.5% and 87% to 95.5%, respectively.

## 4.10.2 Scenario two _after using regularization method

### 4.10.2.1 Result analysis of proposed model by using different dataset split ratio

*Table 4.7 Training and testing accuracy and loss for proposed model Using different dataset split*

| Training split | Test split | Training accuracy | Testing accuracy | Training loss | Testing loss |
|---|---|---|---|---|---|
| 70% | 30% | 97.07% | 96.23% | 0.11% | 0.13% |
| 80% | 20% | 92.90% | 94.70% | 0.19% | 0.14% |

As we can see from Table 4.7, typically, while using the 70/30 ratio of the training, we saw higher performance on the accuracy value than when using the 80/20 ratio.

### 4.10.2.2 Result analysis of proposed model by using various activation mechanisms

Table 4.8 below uses classification accuracy metrics expressed as a percentage for the train and test data separately to show the outcome of an experiment utilizing the proposed model with various activation functions.

*Table 4.8 result of proposed model using different activation function*

| Activation function | Training accuracy | Testing accuracy | Training loss | Testing loss |
|---|---|---|---|---|
| softmax | 97.07.8% | 96.23% | 0.11% | 0.13% |
| sigmoid | 92.90% | 94.70% | 0.19% | 0.14% |

As seen in Table 4.8, it is preferable to use the softmax activation function for multiclass classification than the sigmoid, which is more commonly used for binary classification issues. Using the output layer activation function of Softmax with training and testing dataset ratio of 7:3, the suggested model successfully identifies the provided image with a mean training accuracy of 97.07% and a mean test accuracy of 96.23%.

The accuracy of the model at various epochs is shown by the training and validation accuracy curves in figure 4.8 below. The accuracy curve for several epochs displays various tendencies. As we see in the figure initially, transcription and validation accuracy rise up to epoch 3 almost linearly. Moreover, the validation accuracy deviates significantly from the training accuracy after epoch 3 and reaches approximately 94.70% accuracy at the final epoch. While training accuracy increases almost linearly up to epoch 6 before deviating significantly from training accuracy and ending at 92.90%. Also, during the training phase, validation accuracy is higher than training accuracy, but the difference becomes minor at the final epoch. Ultimately, the curve near the last epochs becomes more stable (for both training and validation accuracy).

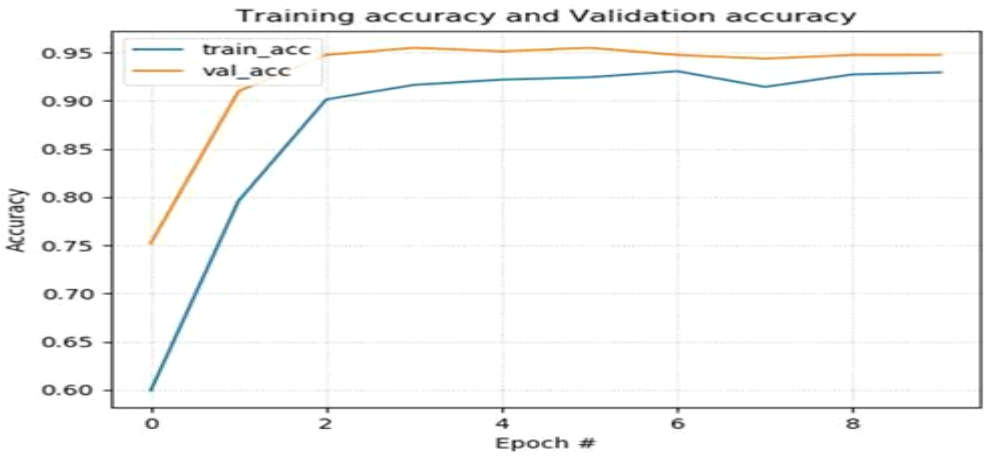



*Figure 4.7 training and validation accuracy of proposed model with sigmoid activation function*

As shown in figure 4.9 below, training and validation losses both sharply decline until epoch 5, when they drop to 0.19. It is evident that the validation loss increased between epochs 5 and 6, then decreased nearly linearly until it reached approximately 0.14 at the final epoch. In contrast, training loss drops off almost linearly up until epoch 6, after which it fluctuates up and down until it hits approximately 0.19 at the last epoch. Throughout the training phase, validation loss is less than training loss, but the difference is negligible. Eventually, as we approach the last epochs, the curves (for both training and validation loss) become more stable.

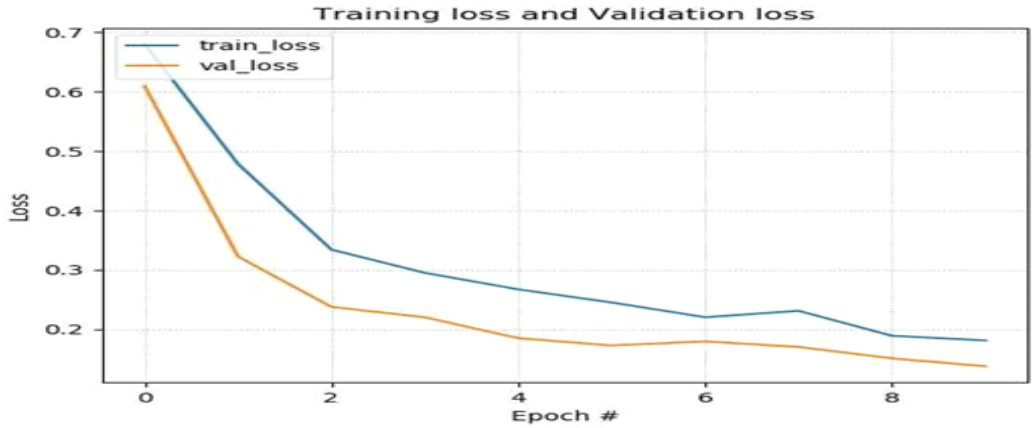



*Figure 4.8 training and validation loss of proposed model with sigmoid activation function*

**Detection and classification report for wheat rust detection model using sigmoid activation function**

The model constructed employing a sigmoid classifier, which is clearly shown in figure 4.10 below, achieves classification accuracy, precision, recall, and an f1_score of 94%,94% and 95%, respectively. This classification accuracy is achieved by training and testing the model without utilizing the regularization approach.

```
12/12 [==============================] - 25s 2s/step
classification_report
              precision    recall  f1-score   support

      health       0.94      0.95      0.94       250
   leaf_rust       0.94      0.94      0.94       135
   stem_rust       0.94      0.94      0.94        73
      yellow       0.94      0.95      0.94       265

    accuracy                           0.94       723
   macro avg       0.94      0.94      0.94       723
weighted avg       0.94      0.94      0.94       723
```

*Figure 4.9 Classification report for proposed model with sigmoid activation function*

As shown in the figure below 4.11, the training and validation accuracy curves show that training accuracy increases nearly linearly up to epoch 5, then oscillates significantly up and down, reaching 97.07% at the final epoch. The validation accuracy increases nearly linearly up to epoch 4, oscillates significantly, and reaches 96.23% at the last epoch. However, starting with epoch 3 and up to the last epoch, both transcription and validation accuracy are comparable, and the difference is negligible.

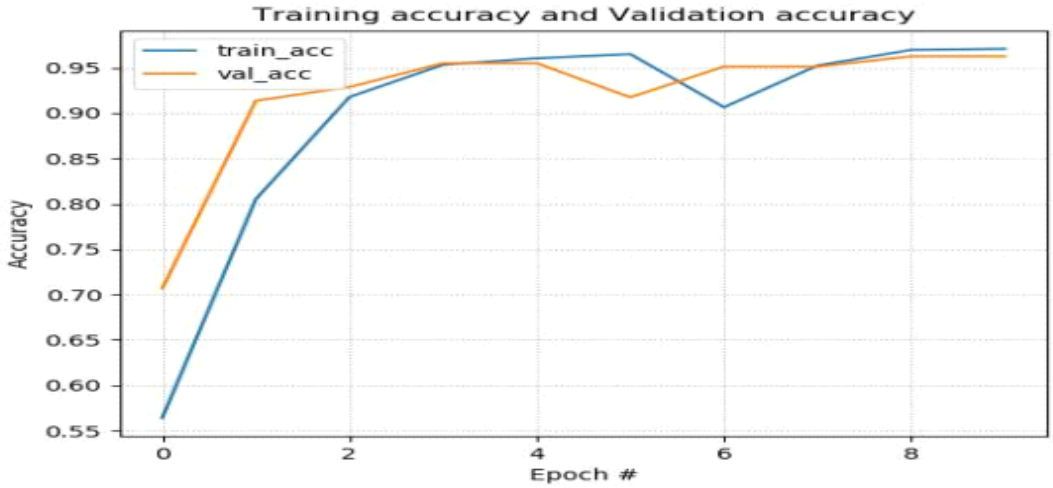



*Figure 4.10 training and validation accuracy of proposed model with softmax activation function*

The figure 4.12 below shows that the first training and validation loss was significant. It is evident that the validation accuracy decreases steadily up to epoch 4 and then fluctuates greatly, reaching 0.13 at the final epoch. While the training loss likewise sharply declines until epoch 5, oscillates up and down, and eventually approaches 0.11, it does so.

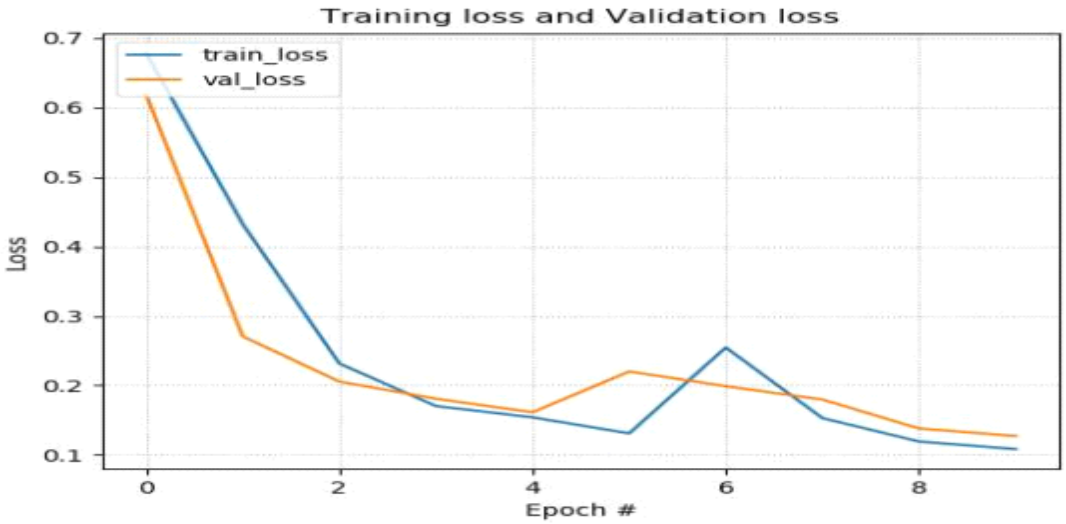



*Figure4.12trainingandvalidationlossofproposedmodel with softmax activation function*

**Detection and classification report for wheat rust detection model using sigmoid activation function**

As clearly shown in figure 4.13 below, the model developed using softmax classifier achieves classification accuracy, precision, and recall and f1- score of 96%. This classification accuracy is obtained when the model is trained and tested after using regularization method.

```
12/12 [==============================] - 25s 2s/step
classification_report
              precision    recall  f1-score   support

      health       0.96      0.96      0.96       250
   leaf_rust       0.96      0.96      0.96       135
   stem_rust       0.96      0.96      0.96        73
      yellow       0.96      0.96      0.96       265

    accuracy                           0.96       723
   macro avg       0.96      0.96      0.96       723
weighted avg       0.96      0.96      0.96       723
```

*Figure 4.13 classification report for proposed model with softmax activation function*

## 4.11 Discussion of result

There were 2538(15318, after augmentation) total images used to train the suggested wheat rust detection model and the pre-trained CNN models. With a unique dataset that wasn't used for model training, the model was successfully tested. The suggested model achieves best performance then models presented in our related works section. Metrics for classification accuracy are used to rate the effectiveness of the models. The mean training accuracy of the proposed wheat rust disease detection model, as shown in the above experiments, is 97.07%. For the suggested wheat rust disease detection models, the mean percentage of testing correctness is 96.23%. We demonstrate that the proposed wheat rust disease detection model has no over fitting problem due to the regularization techniques we used, allowing us to draw the conclusion that the suggested model has higher generalization performance. The experiments of suggested wheat rust disease detection model, yielded mean training loss of 0.11. Here are research questions answered after completion of our thesis.

What are the suitable hyper parameters that help to enhance the performance of the wheat rust disease detection model?

Answer

In this study, we evaluated the performance of wheat rust disease detection and classification using deep learning, and experimented with different hyper parameters, including learning rate, batch size, number of epochs, and regularization strength. Our results showed that a learning rate of 0.000001, a batch size of 64, L2 regularization, dropout and early stopping led to the best performance, achieving an accuracy of 96.23% in wheat rust disease detection and classification model.

What is the impact of different pre-processing techniques (e.g. image resizing, data augmentation) on the accuracy of wheat rust disease detection and classification?

Answer

In this study, we evaluated the impact of different pre-processing techniques, including image resizing and data augmentation, on the accuracy of wheat rust disease detection and classification using deep learning. Our results showed that data augmentation had the most significant impact on the accuracy of wheat rust disease detection and classification by preventing over fitting and improving generalization. Image resizing had a positive impact on the accuracy of the model by standardizing the size of images in the dataset, which can help reduce computational complexity during training and improve the performance of the model.

What are the suitable regularization algorithms for enhancing performance of proposed model?

Answer

In our study we showed the best performance after using different regularization techniques like L2 regularization, dropout, and early stopping. Specifically, our model achieved an accuracy of 96.23% with early stopping, dropout 0.3, and L2 regularization 0.00001. These results suggest that a combination of early stopping, dropout, and L2 regularization can be effective in improving the accuracy of wheat rust disease detection and classification model.

How to evaluate performance of proposed model?

Answer

In our study, we evaluated the performance of the wheat rust disease detection and classification model using accuracy, precision, and recall. We also assessed our model by using the F1 score, which is the most appropriate performance evaluation metric in small datasets because it takes into account both precision and recall, which are important measures in small datasets even if the number of samples in each class is different.

# CHAPTER FIVE

# 5. CONCLUSION AND RECOMMENDATION

## 5.1 Conclusion

At the present time, there are a number of issues with wheat production, including various wheat diseases like unimproved local cultivars and the quality of wheat yield affected by wheat yellow rust, leaf rust, and stem rust, which can result in production losses of up to 100%. Also, the lack of diagnostic equipment in emerging nations like Ethiopia has a difficult challenge on their progress and standard of living. Consequently, it is crucial to adopt technology that is both economical and user-friendly to identify the condition at an early stage. We have developed a wheat rust disease detection and classification model by using a deep learning strategy and utilizing CNN and an image processing technique to make early diagnoses of the diseases. We have provided a wheat rust disease detection model to identify and categories three types of wheat disease and healthy wheat leaves using wheat leaf and stem images as input. To identify wheat leaf and stem diseases, one can utilize the wheat rust disease detection model. This study applied various significant aspects that may have an impact on the model created for the study when discussing the CNN model. To run the tests for the wheat rust disease detection model, an RGB data set, a collection of images with three channels, is used. The RGB image datasets have been augmented using image augmentation techniques. With 50 epochs, the wheat rust disease detection model's accuracy was 86.5 percent; it learned at a rate of 0.000001 before applying the regularization method.

This outcome is enhanced by 94.70% when the model is trained after applying different regularization techniques with a sigmoid activation function, a learning rate of 0.000001, and a 10 epoch number with an 80/20 train-test dataset split.

And which eventually increased to an accuracy of 96.23% after applying regularization technique with softmax activation function, archived better performance when model was trained with softmax activation function, learning rate 0.000001, 10 epochs, and 70/30 train-test split. As we seen in the experiment, regularization technique and learning rate are the most crucial variables that affect the model's effectiveness and accuracy.

## 5.2 Recomendation

The economy of Ethiopia is dominated by agriculture. When identifying numerous plant diseases, image processing and deep learning technologies are crucial, including fruits, vegetables, cereal crops, and other foods. The research conducted in Ethiopia using image processing and deep learning to support the agricultural sector, particularly in disease detection and plant disease prediction, as we discuss in the related work, is minimal; there are already works by some researchers contributing to cases like maize leaf and coffee leaf disease identification and others. As a result, this research might encourage others to continue their work in the field. The facts, algorithms, rules, and feature analysis concepts used in this work can be adopted by researchers who are interested in designing and developing any system linked to plant leaf disease detection or identification with comparable methodologies, with or without alterations. We used a variety of strategies in this study, and the outcome was positive and promising.

Yet, additional research is necessary to create a more accurate diagnosis of the disease affecting wheat crops and produce better results. In future work, a high-dimensional dataset is required for training and testing the suggested model in order to improve it and further boost model accuracy. Only three diseases of wheat plants were taken into account in this research, although there are many more kinds of diseases that can harm other types of food plants.

In order to extract incredibly complex information from the image, we will test the model in the future with a huge number of image datasets with sophisticated configurations by increasing the number of layers and the number of parameters in each layer. Future work should strive to increase wheat rust disease detection and classification accuracy to over 96.23%.

## 5.3 Contribution

In this study, we investigated the use of deep learning algorithms for the detection and classification of plant diseases, specifically focusing on the wheat rust disease. We developed a deep learning model that utilizes a convolutional neural network (CNN) to extract features from images of wheat leaves, and stem. Our contribution to the field of plant disease detection and classification lies in the development of an accurate and efficient deep learning model for wheat rust disease detection. This model can be used by farmers and agricultural researchers to monitor and manage the spread of wheat rust disease, ultimately improving crop yield and reducing the use of pesticides.

We collected the 585 image data from the agricultural research center, which included high-quality healthy wheat images and images affected by 3 different wheat rust diseases. We also obtained 1953 publicly available datasets from online sources, which included wheat crop images from different regions and countries. We combined the datasets and performed preprocessing steps, including image resizing, augmentation, and normalization, to ensure uniformity, compatibility and improve the quality of the data.

# Reference


[1] M. K. Khan, A. Pandey, T. Athar, S. Choudhary, R. Deval, S. Gezgin*, et al.*, "Fusarium head blight in wheat: contemporary status and molecular approaches," *3 Biotech,* vol. 10, pp. 1-17, 2020.

[2] M. Meyer, N. Bacha, T. Tesfaye, Y. Alemayehu, E. Abera, B. Hundie*, et al.*, "Wheat rust epidemics damage Ethiopian wheat production: A decade of field disease surveillance reveals national-scale trends in past outbreaks," *PloS one,* vol. 16, p. e0245697, 2021.

[3] M. Nigus, H. Shimelis, I. Mathew, and S. Abady, "Wheat production in the highlands of Eastern Ethiopia: opportunities, challenges and coping strategies of rust diseases," *Acta Agriculturae Scandinavica, Section B—Soil & Plant Science,* pp. 1-13, 2022.

[4] T. Acharya and A. K. Ray, *Image processing: principles and applications*: John Wiley & Sons, 2005.

[5] D. Hodson, M. Jaleta, K. Tesfaye, C. Yirga, H. Beyene, A. Kilian*, et al.*, "Ethiopia's transforming wheat landscape: tracking variety use through DNA fingerprinting," *Scientific Reports,* vol. 10, p. 18532, 2020.

[6] S. Sood, H. Singh, and S. Jindal, "Rust disease classification using deep learning based algorithm: the case of wheat," *Food Systems Resilience,* p. 197, 2022.

[7] M. Meyer, L. Burgin, M. Hort, D. Hodson, and C. Gilligan, "Large-scale atmospheric dispersal simulations identify likely airborne incursion routes of wheat stem rust into Ethiopia," *Phytopathology,* vol. 107, pp. 1175-1186, 2017.

[8] N. S. Yesuf, S. Getahun, S. Hassen, Y. Alemayehu, K. G. Danu, Z. Alemu*, et al.*, "Distribution, dynamics, and physiological races of wheat stem rust (Puccinia graminis f. sp. tritici) on irrigated wheat in the Awash River Basin of Ethiopia," *PloS one,* vol. 16, p. e0249507, 2021.

[9] M. A. Genaev, E. S. Skolotneva, E. I. Gultyaeva, E. A. Orlova, N. P. Bechtold, and D. A. Afonnikov, "Image-based wheat fungi diseases identification by deep learning," *Plants,* vol. 10, p. 1500, 2021.

[10] S. Dey Sarkar, S. Goswami, A. Agarwal, and J. Aktar, "A novel feature selection technique for text classification using Naive Bayes," *International scholarly research notices,* vol. 2014, 2014.

[11] V. Gorodetsky and V. Samoylov, "Feature extraction for machine learning: logic-probabilistic approach," in *Feature Selection in Data Mining*, 2010, pp. 55-65.

[12] J. Amara, B. Bouaziz, and A. Algergawy, "A deep learning-based approach for banana leaf diseases classification," *Datenbanksysteme für Business, Technologie und Web (BTW 2017)-Workshopband,* 2017.

[13] A. Kamilaris and F. X. Prenafeta-Boldú, "A review of the use of convolutional neural networks in agriculture," *The Journal of Agricultural Science,* vol. 156, pp. 312-322, 2018.

[14] K. Weiss, T. M. Khoshgoftaar, and D. Wang, "A survey of transfer learning," *Journal of Big data,* vol. 3, pp. 1-40, 2016.

[15] T. Zhang, Z. Xu, J. Su, Z. Yang, C. Liu, W.-H. Chen*, et al.*, "Ir-UNet: Irregular Segmentation U-Shape Network for Wheat Yellow Rust Detection by UAV Multispectral Imagery," *Remote Sensing,* vol. 13, p. 3892, 2021.

[16] J. Lu, J. Hu, G. Zhao, F. Mei, and C. Zhang, "An in-field automatic wheat disease diagnosis system," *Computers and electronics in agriculture,* vol. 142, pp. 369-379, 2017.

[17] S. Albawi, T. A. Mohammed, and S. Al-Zawi, "Understanding of a convolutional neural network," in *2017 international conference on engineering and technology (ICET)*, 2017, pp. 1-6.

[18] S. Sood and H. Singh, "An implementation and analysis of deep learning models for the detection of wheat rust disease," in *2020 3rd International Conference on Intelligent Sustainable Systems (ICISS)*, 2020, pp. 341-347.

[19] H. G. Mariam, "Wheat production and research in Ethiopia," 1991.

[20] T. Seemann, "Digital image processing using local segmentation," Monash University, 2002.

[21] K. Moriwaki, I. Koike, T. Sano, T. FUKUNAGA, and K. TANAKA, "Basics and Applications of Digital Image Processing Basics and Applications of Digital Image Processing, 2003."
[22] J. Singla, "Technique of Image Registration in Digital Image processing-a review," *International Journal of Information Technology and Knowledge Management,* vol. 5, pp. 239-243, 2012.
[23] S. Pravinthraja and K. Umamaheswari, "Dec. 2011 Bonfring International Journal of Advances in Image Processing article,"," *Multimodal Biometrics for Improving Automatic Teller Machine Security*.
[24] S. Chatterjee, "What is feature extraction? feature extraction in image processing," ed: Great Learning Blog: Free Resources What Matters to Shape Your Career,(2021 …, 2020.
[25] Y. H. Liu, "Feature extraction and image recognition with convolutional neural networks," in *Journal of Physics: Conference Series*, 2018, p. 062032.
[26] M. Masum and P. Laval, "Fusion-Net: Integration of Dimension Reduction and Deep Learning Neural Network for Image Classification," 2020.
[27] S. Dutta, S. K. Pal, S. Mukhopadhyay, and R. Sen, "Application of digital image processing in tool condition monitoring: A review," *CIRP Journal of Manufacturing Science and Technology,* vol. 6, pp. 212-232, 2013.
[28] K. Pavithradevi, M. Pavithra, and S. Mangayarkani, "Digital Image Processing and Its Applications," *IJRASET-International Journal for Research in Applied Science & Engineering Technology,* vol. 5, 2017.
[29] G. Thomas, "OUTLINING A RELEVANT UNDERGRADUATE COURSE ON COMPUTER VISION," *Proceedings of the Canadian Engineering Education Association (CEEA),* 2012.
[30] P. Thamilselvan and J. Sathiaseelan, "A comparative study of data mining algorithms for image classification," *International Journal of Education and Management Engineering,* vol. 5, pp. 1-9, 2015.
[31] D. Roopa and S. Bose, "Leaf's Diseases and Its Characteristics Visualization using Augmented Reality," in *2022 1st International Conference on Computational Science and Technology (ICCST)*, 2022, pp. 1019-1024.
[32] X. Feng, Y. Jiang, X. Yang, M. Du, and X. Li, "Computer vision algorithms and hardware implementations: A survey," *Integration,* vol. 69, pp. 309-320, 2019.
[33] K. K. Singh and A. Singh, "A study of image segmentation algorithms for different types of images," *International Journal of Computer Science Issues (IJCSI),* vol. 7, p. 414, 2010.
[34] J. Castañón, "Machine learning methods that every data scientist should know," *Consultado em Outubro,* vol. 16, 2019.
[35] D. Fumo, "Types of machine learning algorithms you should know," *towards data science,* vol. 15, 2017.
[36] A. Ulges, "Neural Networks I," 1991.
[37] M. Galety, F. H. Al Mukthar, R. J. Maaroof, and F. Rofoo, "Deep neural network concepts for classification using convolutional neural network: A systematic review and evaluation," 2021.
[38] L. Habashy, "Comparing Machine Learning," 2020.
[39] F. Raheem and N. Iqbal, "and Machine Learning for the Industrial Internet of Things (IIoT)," *Industrial Internet of Things: Technologies and Research Directions,* p. 1, 2022.
[40] D. C. Jakobsen, "Visual Analysis and Personalisation in Advertisement," The University of Bergen, 2022.
[41] R. Saxena, "How decision tree algorithm works," *Dataaspirant,* 2017.
[42] J. Michael and J. Garbade, "Understanding k-means clustering in machine learning," Retrieved 2019-10-312019.
[43] X. J. Zhu, "Semi-supervised learning literature survey," 2005.

[44] S. Sharma, "Activation functions in neural networks, 2017," *URL: https://towardsdatascience. com/activation-functions-neural-networks-1cbd9f8d91d6*.
[45] J. Daunizeau, "Semi-analytical approximations to statistical moments of sigmoid and softmax mappings of normal variables," *arXiv preprint arXiv:1703.00091,* 2017.
[46] N. Basta, "The Differences between Sigmoid and Softmax Activation Functions," *Retrieved September,* vol. 16, p. 2021, 2020.
[47] S. Sharma, "Activation Functions in Neural Networks, Sept. 2017," *URL https://towardsdatascience. com/activation-functions-neural-networks-1cbd9f8d91d6*.
[48] K. Supakkul, K. Chang, R. Beauchamp, and M. Tiwari, "Machine Learning Classification of Bacterial vs Fungal Keratitis from Photographs."
[49] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*: MIT press, 2016.
[50] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature,* vol. 521, pp. 436-444, 2015.
[51] J. Patterson and A. Gibson, *Deep learning: A practitioner's approach*: " O'Reilly Media, Inc.", 2017.
[52] M. Gurucharan, "Basic cnn architecture: Explaining 5 layers of convolutional neural network," *URL: https://www. upgrad. com/blog/basic-cnn-architecture,* 2020.
[53] E. B. Paulos and M. M. Woldeyohannis, "Detection and Classification of Coffee Leaf Disease using Deep Learning," in *2022 International Conference on Information and Communication Technology for Development for Africa (ICT4DA)*, 2022, pp. 1-6.
[54] A. J. Belay, A. O. Salau, M. Ashagrie, and M. B. Haile, "Development of a chickpea disease detection and classification model using deep learning," *Informatics in Medicine Unlocked,* vol. 31, p. 100970, 2022.
[55] T. Aboneh, A. Rorissa, R. Srinivasagan, and A. Gemechu, "Computer vision framework for wheat disease identification and classification using Jetson GPU infrastructure," *Technologies,* vol. 9, p. 47, 2021.
[56] L. Li, S. Zhang, and B. Wang, "Plant disease detection and classification by deep learning—a review," *IEEE Access,* vol. 9, pp. 56683-56698, 2021.
[57] B. M. Abuhayi and A. A. Mossa, "Coffee disease classification using Convolutional Neural Network based on feature concatenation," *Informatics in Medicine Unlocked,* vol. 39, p. 101245, 2023.
[58] S. Mishra, R. Sachan, and D. Rajpal, "Deep convolutional neural network based detection system for real-time corn plant disease recognition," *Procedia Computer Science,* vol. 167, pp. 2003-2010, 2020.
[59] !!! INVALID CITATION !!!
[60] A. Chand and M. Arora, "Detection of Rust Disease in Wheat Crop Using Convolutional Neural Networks via Transfer Learning Technique," in *Advances in Interdisciplinary Engineering*, ed: Springer, 2021, pp. 499-507.
[61] G. Shrestha, M. Das, and N. Dey, "Plant disease detection using CNN," in *2020 IEEE Applied Signal Processing Conference (ASPCON)*, 2020, pp. 109-113.
[62] D. Kumar and V. Kukreja, "N-CNN based transfer learning method for classification of powdery mildew wheat disease," in *2021 International Conference on Emerging Smart Computing and Informatics (ESCI)*, 2021, pp. 707-710.
[63] S. Sood, H. Singh, and S. Jindal, "Rust Disease Classification Using Deep Learning Based Algorithm: The Case of Wheat," 2022.
[64] M. Sardogan, A. Tuncer, and Y. Ozen, "Plant leaf disease detection and classification based on CNN with LVQ algorithm," in *2018 3rd international conference on computer science and engineering (UBMK)*, 2018, pp. 382-385.
[65] D. Kumar and V. Kukreja, "An Instance Segmentation Approach for Wheat Yellow Rust Disease Recognition," in *2021 International Conference on Decision Aid Sciences and Application (DASA)*, 2021, pp. 926-931.

[66] I. U. Haq, R. Mumtaz, M. Talha, Z. Shafaq, and M. Owais, "Wheat Rust Disease Classification using Edge-AI," in *2022 2nd International Conference on Artificial Intelligence (ICAI)*, 2022, pp. 58-63.
[68] S. Demeyer, "Research methods in computer science," in *ICSM*, 2011, p. 600.
[69] O. Spirin, V. Oleksiuk, O. Oleksiuk, and S. Sydorenko, "The group methodology of using cloud technologies in the training of future computer science teachers," 2018.
[70] A. Maedche, S. Gregor, and J. Parsons, "Mapping design contributions in information systems research: the design research activity framework," *Communications of the Association for Information Systems,* vol. 49, p. 12, 2021.
[71] M. Lorentzon, "Feature extraction for image selection using machine learning," ed, 2017.
[72] J. SHUBHAM, "An overview of regularization techniques in deep learning," *Analytics Vidhya. Retrieved on https://www. analyticsvidhya. com/blog/2018/04/fundamentals-deep-learning-regularization-techniques/2018 on July,* 2019.
[73] P. Panwar, G. Gopal, and R. Kumar, "Image Segmentation using K-means clustering and Thresholding," *Image,* vol. 3, pp. 1787-1793, 2016.

# Appendix

Appendix A

Photo images of wheat

**Appendix B implementation of model**

```
In [1]: import os
import cv2
import glob
import pandas as pd
import numpy as np
import keras
import tensorflow as tf
import matplotlib.pyplot as plt
from keras.layers import Dense
from keras.models import Sequential
from keras.preprocessing import image
from keras.optimizers import Adam
from keras.models import Model
from time import time
import sklearn.metrics as metrics
from keras.callbacks import ReduceLROnPlateau
from keras.preprocessing.image import ImageDataGenerator
from keras.layers import Convolution2D,Dense,MaxPool2D,Activation,Dropout,Flatten
from keras.layers import Input, Add, Dense, Activation, ZeroPadding2D, BatchNormalization, Flatten, Conv2D, AveragePooling2D, Ma
from keras.regularizers import l2
from sklearn.metrics import classification_report
```

Source code to import necessary packages (python libraries)

```
In [5]: def get_files(directory):
  if not os.path.exists(directory):
    return 0
  count=0
  # crawls inside folders
  for current_path,dirs,files in os.walk(directory):
    for dr in dirs:
      count+= len(glob.glob(os.path.join(current_path,dr+"/*")))
  return count
train_dir ="C:/Users/3090/Desktop/anchinesh/train"
test_dir= "C:/Users/3090/Desktop/anchinesh/test"
```

```
In [6]: train_samples =get_files(train_dir)
#to get tags
num_classes=len(glob.glob(train_dir+"/*"))
#test file image count
test_samples=get_files(test_dir)
print(num_classes,"Classes")
print(train_samples,"train images")
print(test_samples,"test images")
```

```
4 Classes
1815 train images
723 test images
```

Source code to list fillies in specific folder

```
In [8]: model = Sequential()
        model.add(Conv2D(32, (5, 5),input_shape=input_shape,activation='relu',name="conv2d_1"))
        model.add(MaxPooling2D(pool_size=(3, 3),name="max_pooling2d_1"))

        model.add(Conv2D(32, (3, 3),activation='relu',name="conv2d_2"))
        model.add(MaxPooling2D(pool_size=(2, 2),name="max_pooling2d_2"))

        model.add(Conv2D(64, (3, 3),activation='relu',name="conv2d_3"))
        model.add(MaxPooling2D(pool_size=(2, 2),name="max_pooling2d_3"))

        model.add(Conv2D(64, (5, 5),activation='relu',name="conv2d_4"))
        model.add(MaxPooling2D(pool_size=(2, 2),name="max_pooling2d_4"))

        model.add(Conv2D(128, (3, 3),activation='relu',name="conv2d_5"))
        model.add(MaxPooling2D(pool_size=(2, 2),name="max_pooling2d_5"))


        model.add(Conv2D(256, (1, 1),activation='relu',name="conv2d_6"))
        model.add(MaxPooling2D(pool_size=(2, 2),name="max_pooling2d_6"))

        model.add(Flatten(name="flatten_1"))
        model.add(Dense(512,activation='relu'))
        model.add(Dropout(0.25))
        model.add(Dense(128,activation='relu'))
        model.add(Dense(num_classes,activation='softmax'))
        model.summary()
```

Source code for proposed model before using regularization algorithms

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 220, 220, 32)      2432

max_pooling2d_1 (MaxPooling  (None, 73, 73, 32)        0
2D)

conv2d_2 (Conv2D)            (None, 71, 71, 32)        9248

max_pooling2d_2 (MaxPooling  (None, 35, 35, 32)        0
2D)

conv2d_3 (Conv2D)            (None, 33, 33, 64)        18496

max_pooling2d_3 (MaxPooling  (None, 16, 16, 64)        0
2D)

conv2d_4 (Conv2D)            (None, 12, 12, 64)        102464

max_pooling2d_4 (MaxPooling  (None, 6, 6, 64)          0
2D)

conv2d_5 (Conv2D)            (None, 4, 4, 128)         73856

max_pooling2d_5 (MaxPooling  (None, 2, 2, 128)         0
2D)

conv2d_6 (Conv2D)            (None, 2, 2, 256)         33024

max_pooling2d_6 (MaxPooling  (None, 1, 1, 256)         0
2D)

flatten_1 (Flatten)          (None, 256)               0

dense (Dense)                (None, 512)               131584

dropout (Dropout)            (None, 512)               0

dense_1 (Dense)              (None, 128)               65664

dense_2 (Dense)              (None, 4)                 516

=================================================================
Total params: 437,284
Trainable params: 437,284
Non-trainable params: 0
```

Summary of proposed model before applying regularization method

```
In [9]: model = Sequential()
        model.add(Conv2D(32, (5, 5),input_shape=input_shape,activation='relu',name="conv2d_1"))
        model.add(MaxPooling2D(pool_size=(3, 3),name="max_pooling2d_1"))

        model.add(Conv2D(32, (3, 3),kernel_regularizer=l2(0.0001), bias_regularizer=l2(0.0001),activation='relu',name="conv2d_2"))
        model.add(MaxPooling2D(pool_size=(2, 2),name="max_pooling2d_2"))

        model.add(Conv2D(64, (3, 3),kernel_regularizer=l2(0.0001), bias_regularizer=l2(0.0001),activation='relu',name="conv2d_3"))
        model.add(MaxPooling2D(pool_size=(2, 2),name="max_pooling2d_3"))

        model.add(Conv2D(64, (5, 5),kernel_regularizer=l2(0.0001), bias_regularizer=l2(0.0001),activation='relu',name="conv2d_4"))
        model.add(MaxPooling2D(pool_size=(2, 2),name="max_pooling2d_4"))

        model.add(Conv2D(128, (3, 3),kernel_regularizer=l2(0.0001), bias_regularizer=l2(0.0001),activation='relu',name="conv2d_5"))
        model.add(MaxPooling2D(pool_size=(2, 2),name="max_pooling2d_5"))
        model.add(Dropout(0.25))
        model.add(Conv2D(256, (1, 1),kernel_regularizer=l2(0.0001), bias_regularizer=l2(0.0001),activation='relu',name="conv2d_6"))
        model.add(MaxPooling2D(pool_size=(2, 2),name="max_pooling2d_6"))
        model.add(Dropout(0.25))
        model.add(Flatten(name="flatten_1"))
        model.add(Dense(512,activation='relu',kernel_regularizer=l2(0.0001),
                        bias_regularizer=l2(0.001)))
        model.add(Dropout(0.25))

        model.add(Dense(512,activation='relu'))
        model.add(Dense(4, activation='softmax',kernel_regularizer=l2(0.0001),
                        bias_regularizer=l2(0.0001)))
        callback = tf.keras.callbacks.EarlyStopping(monitor='loss', patience=5)

        model.summary()
```

Source code for proposed model with L2 regularization

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_1 (Conv2D)           (None, 220, 220, 32)      2432

 max_pooling2d_1 (MaxPooling  (None, 73, 73, 32)       0
 2D)

 conv2d_2 (Conv2D)           (None, 71, 71, 32)        9248

 max_pooling2d_2 (MaxPooling  (None, 35, 35, 32)       0
 2D)

 conv2d_3 (Conv2D)           (None, 33, 33, 64)        18496

 max_pooling2d_3 (MaxPooling  (None, 16, 16, 64)       0
 2D)

 conv2d_4 (Conv2D)           (None, 12, 12, 64)        102464

 max_pooling2d_4 (MaxPooling  (None, 6, 6, 64)         0
 2D)

 conv2d_5 (Conv2D)           (None, 4, 4, 128)         73856

 max_pooling2d_5 (MaxPooling  (None, 2, 2, 128)        0
 2D)

 dropout (Dropout)           (None, 2, 2, 128)         0

 conv2d_6 (Conv2D)           (None, 2, 2, 256)         33024

 max_pooling2d_6 (MaxPooling  (None, 1, 1, 256)        0
 2D)

 dropout_1 (Dropout)         (None, 1, 1, 256)         0

 flatten_1 (Flatten)         (None, 256)               0

 dense (Dense)               (None, 512)               131584

 dropout_2 (Dropout)         (None, 512)               0

 dense_1 (Dense)             (None, 512)               262656

 dense_2 (Dense)             (None, 4)                 2052

=================================================================
Total params: 635,812
Trainable params: 635,812
Non-trainable params: 0
_________________________________________________________________
```

Summary of proposed model with regularization technique

```
In [*]: model.compile(optimizer='adam',loss = 'categorical_crossentropy',metrics=['accuracy'])
history1 = model.fit(
    train_generator,#egitim verileri
    steps_per_epoch=None,
    epochs=100,
    validation_data=validation_generator,
    validation_steps=None,
    verbose=1,
    callbacks=[ReduceLROnPlateau(monitor='val_loss', factor=0.3,patience=5, min_lr=0.000001)],
    shuffle=True
)
```

Sample source code to compile and triaging the proposed model

```
In [59]: from keras.models import Sequential
from keras.layers import Dense
import matplotlib.pyplot as plt
import numpy

print(history1.history.keys())

plt.plot(history1.history['accuracy'])
plt.plot(history1.history['val_accuracy'])
plt.title('Training accuracy and validation accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch #')
plt.legend(['train_acc', 'val_acc'], loc='upper left')
plt.minorticks_on()
plt.grid(True)
plt.show()
# summarize history for loss
plt.plot(history1.history['loss'])
plt.plot(history1.history['val_loss'])
plt.title('Training loss and validation loss')
plt.ylabel('loss')
plt.xlabel('epoch #')
plt.legend(['train_loss', 'val_loss'], loc='upper left')
plt.minorticks_on()
plt.grid(True)
plt.show()
```

Source code to plotting accuracy and loss graph for proposed model

```
In [62]: test_generator.reset()
predictions = model.predict(test_generator, steps=len(test_generator))
y = np.argmax(predictions, axis=1)

print('classification_report')
cr = classification_report(y_true=test_generator.classes, y_pred=y, target_names=test_generator.class_indices)
print(cr)
```

Sample source code show classification report for proposed model