HAWASSA UNIVERSITY

INSTITUTE OF TECHNOLOGY

FACULTY OF INFORMATICS

DEPARTMENT OF COMPUTER SCIENCE

CABBAGE DISEASE CLASSIFICATION USING CONVOLUTIONAL NEURAL NETWORK

BY

SAMRAWIT FELEKE

May, 2024

HAWASSA UNIVERSITY IoT, HAWASSA, ETHIOPIA

.

HAWASSA UNIVERSITY

INSTITUTE OF TECHNOLOGY

FACULTY OF INFORMATICS

DEPARTMENT OF COMPUTER SCIENCE

CABBAGE DISEASE CLASSIFICATION USING CONVOLUTIONAL NEURAL NETWORK

M.Sc. in Computer Science

By: Samrawit Feleke

Advisor: Dr. Degif Teka (PhD)

A THESIS SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCES, FACULTY OF INFORMATICS, HU-IoT, SCHOOL OF GRADUATE STUDIES HAWASSA UNIVERSITY HAWASSA, ETHIOPIA

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

May, 2024

.

_____	_____
Name of the Student	Signature and Date

Approved by:

_____	_____
Name of the Advisor	Signature and Date
_____	_____
Name of the Co-advisor	Signature and Date
_____	_____
Name of the school head	Signature and Date
_____	_____
Name of the Postgraduate head	Signature and Date

(Postgraduate/ Department stamp)

.

Acknowledgment

I want to express my deepest gratitude to my advisors for their guidance and support throughout the completion of this master's thesis. I am also grateful to all Hawassa University Agricultural College professors, laboratory experts' colleagues, and friends who have provided valuable feedback and assistance. Finally, I would like to thank my family for their continuous love and encouragement. This thesis would not have been possible without the help and support of all these individuals.

.

Contents

.

.

.

.

List of Table

.

List of Abbreviation

CSA	Central statistical agency
CNN	convolutional neural networks
NLP	Naturale language processing
AI	Artificial Intelligence
ML	Machine learning
DL	deep learning
ANN	Artificial neural network
RNN	Recurrent neural networks
LSTM	Long short-term memory
GAN	Generative adversarial networks
2D	Two dimensional
ReLU	Rectified linear unit
FC	Fully connected
GS	Grid search
RS	Random Search
TP	True positive
FN	False-negative
FP	False positive
TN	True negative
IoU	Intersection over union
mAP	mean average precision
AP	average precision
AUC	area under the curve
SVM	support vector machine
RF	random forest
1D	CNN one-dimensional convolution network
PIL	Python imaging library
OpenCV	open-source computer vision

.

Abstract

Cabbage is a widely cultivated vegetable crop susceptible to various diseases, which can significantly reduce crop yield and quality. To effectively monitor these diseases, accurate and timely classification is essential. This study proposes cabbage plant disease classification using convolutional neural networks (CNN). The proposed method involves preprocessing input images, extracting relevant features using a developed CNN model, and accurately classifying diseases affecting cabbage plants. The cabbage leaf image dataset was collected from Sidama region Hawassa zuriya deneba and east shewa zone of the Oromia region dugda woreda meki town cabbage production areas in Ethiopia. The performance of the proposed method is evaluated on 5700 datasets of cabbage plant images with five types of diseases, and one healthy cabbage class such as Alternaria leaf spot, anthracnose, black rot, cabbageworm, downy mildew, and healthy cabbage. We used the data augmentation method to expand the training dataset. 70% of the dataset was used for training, and the remaining 30% was used for testing and validation. We used existing research to compare the proposed model and obtained better results. The proposed model is evaluated using certain metrics such as precision, recall, F1-score, and accuracy. 96.80% of precision, 96.65% of recall, and 96.67% of F1-score. The proposed model achieved 99.53% training accuracy and 98.51% test accuracy for classified experiment results. The results show that the proposed approach can provide accurate and efficient disease diagnosis for cabbage plants, enabling timely and targeted intervention to prevent crop loss.

Keywords: Cabbage, Cabbage disease, CNN

.

CHAPTER ONE

1. INTRODUCTION

The study's background, motivation, statement of the problem, objective, scope, limitations, and lastly, the organization of the study are all covered in this chapter.

1.1. Background of the study

Plant disease is an abnormal condition in plants caused by pathogens such as fungi, bacteria, viruses, nematodes, or environmental factors. It can result in alterations in the growth, or functioning of the plant, leading to decreased yield or even death of the plant. In this case, accurate identification and treatment of diseases requires appropriate disease analysis. Cabbage plants are affected by different diseases. Due to the farmers' and agriculture personnel's lack of knowledge, the classification of cabbage plant diseases becomes difficult. The use of disease classification is beneficial for identifying a cabbage plant at different stages.[1] Cabbage is a brassicaceae family flowering plant species. The next paragraph shows different types of brassicaceae family flowering plant species.

The Brassicaceae family flowering plant species, including cabbage, is Brassica oleracea. It is commonly referred to as wild cabbage. Numerous cultivated vegetables, such as Brussels sprouts, kohlrabi, kale, cauliflower, broccoli, and cabbage, have this species as their ancestor. [2] A particular cultivar of cabbage known as white cabbage or green cabbage is Brassica oleracea var. capitata f. alba. Brassica oleracea var. capitata f. alba is the scientific name of cabbage. It belongs to the family Brassicaceae. The most popular variety of cabbage to grow is green cabbage. Thick leaves are placed closely together on the huge, round cabbage head. A cabbage head's upper leaves are usually bright green, turning white or light green as they approach the center. [2] Cabbage is a green leafy vegetable that ranges from 500 grams to 1,000 grams and can be grown in three months. Cabbage is generally available in three cultivars: green, red, and savoy. Green cabbage is the most common and has smooth-leafed, firm-headed green leaves. Unlike green cabbage, red cabbage has smooth-leafed purple leaves and is less common than green cabbage. [3] Figure 1.1 shows Brassica oleracea var. capitata f. alba (green cabbage).

.

Figure1.1 Brassica oleracea var. capitata f. alba (green cabbage)

Both humans and animals can eat cabbage because of its nutritional worth. In 100 grams of cabbage, there is 9.8% water, 1.5 mg protein, 55.0 mg calcium, and 0.8 mg iron. Cabbage also contains minimal levels of vitamin A, iron, and riboflavin, among other minerals. [3]. Due to its antioxidant, anti-inflammatory, and anti-bacterial properties, cabbage is broadly utilized in conventional medicine to alleviate signs and symptoms related to gastrointestinal disorders[2].

Cabbage is the fifth most important vegetable of member Cruciferae family. It is also an important and varied collection of vegetables that are farmed all over the world. From the earliest times of antiquity, cabbage was domesticated and utilized for human use. It originated in Western Europe and along the coast of the North Sea.[2]

Ethiopia's agroclimatic conditions are favorable for producing cabbage for the fresh market. [2]. According to the Central Statistical Agency (CSA) 2017/2018 annual report, after red pepper, cabbage is the second most important vegetable crop in Ethiopia in terms of both production and area covered. There are many types of cabbage. However, this study focuses on a cabbage variety known as ‘Tikil Gomen’ or “□□□ □□□” in Amharic.

Although it has a good yield, it has some major types of diseases and pests occur when it comes to the cultivation of ‘Tikil Gomen’ in Ethiopia. If action is not taken, it can cause serious damage to the cabbage farm. As we have observed in the area of Sidama region Hawassa zuriya Deneba woreda and East Shewa Zone of the Oromia region Dugda woreda Meki town ‘Tikil Gomen’ which are high production areas of Ethiopia, we found some common cabbage diseases with the support

.

of an expert after data collection that are Alternaria leaf spot, Anthracnose, black rot, downy mildew, cabbage worm.

For the classification of these diseases, both domain experts and farmers use traditional methods. Since it is labor-intensive and ineffective to reach cabbage farms in these situations, we require a digital system for the identification and classification of such plant diseases. The most popular technologies for classifying and identifying plant diseases are computer vision and image analysis. As a result, deep learning, computer vision, and machine learning algorithms are used in most current research to identify diseases in plant images. Deep learning techniques include using convolutional neural networks (CNN) for image classification, object detection, and semantic segmentation. Plant disease identification with CNN-based deep learning has been a popular research area.

Therefore, the implementation of the model in the sector will have paramount importance in facilitating activities such as economic, social, and ecologic development in the country by increasing efficiency in both the quality and quantity of cabbage crop production, sustaining the dependability of customer preferences, and preserving the ecology.

In addition, this study is to enable precision agriculture, which means farming practices have become targeted and focused on specific areas rather than a broad field approach using advanced technology. So fast and reliable cabbage disease classification methods enable precision intervention and allow farmers to take necessary action such as applying targeted treatment that reduces unnecessary chemical usage, removing infected cabbage, and preventing further damage.

1.2. Motivation

Cabbage is recognized for its high vitamin, mineral, and dietary fiber content. For both domestic usage and financial gain, Ethiopians typically cultivate cabbage in the country's mid- and high-altitude regions. Ethiopia was the 75th greatest exporter of cabbage in the world in 2021 with $339k in exports. Ethiopia's top export markets for cabbages are Djibouti ($337k), Nigeria ($818), Qatar ($301), Ireland ($220), and the United Arab Emirates ($711).[3]

Although cabbage is used as a source of food, income, employment opportunities, and foreign exchange, its productivity and production are affected by several issues. Low productivity and quality of cabbage can be attributed to several factors, including inadequate agronomic techniques,

.

inadequate insect and disease management and classification systems, and a lack of varieties suitable for the various agroecology in the nation.[4]

The main motivation of this thesis stems from the classification of cabbage diseases using traditional techniques have been a persistent challenge. Researchers have explored some techniques using deep learning and image processing to improve the accuracy of cabbage disease classification. However, the area still requires further research improvement, and the practice is less common in Ethiopia. Therefore, the implementation of image technology in the classification of disease will be of paramount importance in increasing the yield of the cabbage crop

1.3. Statement of the problem

Cabbage crops are susceptible to various diseases, which can significantly impact yield and quality. Manual disease detection by farmers is time-consuming and prone to errors. Therefore, the objective of this thesis is to develop an accurate and efficient CNN-based model capable of classifying cabbage plant images into specific disease categories. The model will aid in early disease classification, allowing timely intervention and improved crop management.

Nowadays, experts identify cabbage plant diseases with the naked eye. According to the interview made with experts: first, when they plant the cabbage, second when the cabbage grows it may be infected by diseases. If experts and farmers observe such a disease, they may treat it using a treatment mechanism. Also, whenever they are in doubt about the diseases that occur, they take samples from diseased plants and test them to identify them through a laboratory. The challenge in the current disease identification mechanism is that they iterate until they are sure about the type of diseases tested in the laboratory. The process usually takes much time, is prone to error, and is inefficient, because of which the disease affects cabbage production. This in turn decreases productivity putting agriculture in harm.

According to Hawassa University plant disease researchers, the following diseases commonly occur in production areas: The most devastating disease affecting the production and productivity of cabbage is Alternaria leaf spot. This disease is caused by two species of fungi, Alternaria brassicicola and A. brassicae. It causes yellow spots that grow larger and develop rings around them like a target or bull's eye The centers of the tissue may come out as it dies, leaving holes in the leaves. The spots come together to create large areas of dead tissue. Bacterial leaf spot disease

.

is caused by two different species of bacteria, Pseudomonas syringae pv. maculicola, it causes spots to form on the leaves that start small, becoming dark brown or purple as they increase in size. These spots can join together to form angular lesions, which make the leaves look ragged. Clubroot disease is caused by a soil-borne fungus called Plasmodiophora brassicae**.** Downy mildew disease is caused by a fungus-like organism called Peronospora parasitica. It causes the yellowing and wilting of leaves and a grayish-white mold on the undersides of leaves**.** White mold disease is caused by a fungus called Sclerotinia sclerotiorum. It causes white, fluffy growth on stems and leaves that eventually turn brown and die off. Therefore, the objective of this research is to build a model that classifies cabbage diseases from input cabbage images using CNN deep-learning technologies.

Although there have been several studies on disease classification in plants using different machine learning techniques, there exists a research gap in the application of deep learning specifically for cabbage disease classification. Most existing studies focus on general plant diseases or specific crop diseases, but very few have explored deep-learning techniques for cabbage disease classification. There is a lack of research specifically on the classification of cabbage diseases using deep learning.

Despite these previous studies, there are still some research gaps and limitations for further investigation in cabbage disease classification using deep learning. These gaps include:

Limited availability of datasets: most existing studies in this area have used small or limited datasets. To improve the accuracy and robustness of the models, our study focuses on gathering larger and more diverse datasets for training and validation.

Lack of temporal analysis: cabbage diseases may exhibit variations over time, such as different stages. Previous studies may not have adequately considered the temporal aspect of disease classification.

Overall, while some progress has been made in cabbage disease classification using deep learning, there are still research gaps that need to be addressed to achieve more accurate and robust disease classification systems.

.

The current study attempts to answer the following **research questions:**

- **Research question 1:** Is it possible to classify cabbage diseases using CNN?
- **Research question 2:** To what extent does the model perform in the classification of cabbage disease?

1.4. Objective

1.4.1. General objective

The general objective of this thesis is to research and develop a model that can automatically detect and classify cabbage plant diseases based on deep learning.

1.4.2. Specific objective

The specific objectives of this research work are:

- Explore the different types of cabbage diseases from experts and literature review.
- Collect dataset from a field of farm
- To preprocess the data
- To design a deep CNN model for cabbage disease classification.
- To train the CNN deep neural network architecture
- To evaluate the performance of the model with quality metrics and existing work

1.5. Significance of the study

It is generally agreed upon that technology has become an essential aspect of our life and its use is unavoidable. Because of this, we need to use technology to raise our standard of living. One such technology that is currently being utilized for the classification of Cabbage diseases from cabbage leaf, stem, and root images is deep learning and machine learning. This research has enabled agriculture experts to recognize the significance of image processing in agriculture, thereby facilitating the identification of cabbage disease through image processing. Early classification of the disease can prevent its spread and enable experts to take timely action to prevent severe damage. This, in turn, can lead to improved quality of cabbage production. The study is designed and develops a model that automatically detects the disease, resulting in benefits such as improv cabbage quality, reduced time consumption, and reduced labor expenses.

.

1.6. Scope and limitation of the study

This study did not consider those diseases affecting other Brassicaceae families is out of the scope of the study. It is specifically limited to green cabbage (Brassica oleracea var. capitata f. alba) 'Tikil Gomen' or "□□□ □□□." As input, both infected and healthy cabbage leaves are used for training and constructing the model. The study uses CNN architecture for learning and cabbage disease detection. This study exclusively addressed six label-class diseases of cabbage leaves.

1.7. Organization of the thesis

This thesis is organized as follows: Chapter One introduces the statement of the problem, the objective, scope, significance, and methodology of the study. Chapter two presents' reviews of concepts and methods relevant to the thesis, reviews of different journals and thesis reports, and all issues related to cabbage, cabbage leaf diseases, image processing, machine learning, deep learning, and so on. Also, related works are reviewed to show the extent to which the problem is solved. Chapter three discusses the proposed architecture of the study; in addition, methods are discussed. Chapter four discusses the experimentation and results of the proposed model in detecting cabbage disease with analysis and interpretation. Finally, chapter five concludes the thesis by drawing conclusions and recommending some future work.

.

CHAPTER TWO

2. LITERATURE REVIEW

This review aims to provide an overview and understanding of the current state of knowledge and research in the field, including the methodologies, techniques, and technologies used for detecting cabbage diseases. The literature review here is used to identify gaps in the existing knowledge, highlight areas for further research, and provide a foundation for developing new and improved disease-classification methods for cabbage plants.

2.1. Cabbage

A literature review on cabbage involves examining previously published research and academic articles regarding the topic. In this context, a brief description of cabbage would include its botanical classification, nutritional content, and common varieties.

Cabbage (Brassica oleracea var. capitata) originated in the eastern Mediterranean regions from a wild non-heading type, 'Cole wart' (Brassica oleracea var. Sylvestris) where it is still found in Denmark, North-Western France, and Eastern England. Cultivated for a long time, this vegetable crop has a long history. The Greeks regarded it as a great harvest of vegetables.[5]

Cabbage is a member of the Brassicaceae family, which is in the genus Brassica. Scientifically speaking, it is called Brassica oleracea var. capitata. The versatility and health benefits of cabbage have led to its widespread cultivation and consumption worldwide. Its compact head is formed by closely packed leaves that give it an elongated or spherical shape. Depending on the cultivar, cabbage can have colors ranging from bright green to deep purple.[5]

In terms of nutrition, cabbage is high in vital vitamins and minerals but low in calories and carbs. It has significant levels of fiber, folate, manganese, and the vitamins K, C, and B6. Cabbage is thought to offer multiple health benefits, including potential anti-inflammatory and anticancer qualities, because of its nutritional density and different antioxidants.[5]

Green cabbage, red cabbage, and savoy cabbage are some of the common kinds of cabbage. Because of their somewhat varied qualities, flavors, and textures, each variety can be used in several culinary applications. For example, red cabbage works well in salads or pickles, whereas green cabbage is frequently used in recipes like salads.[6]

.

China is the world's top producer and exporter of cabbage. It produces 34,151,665 tons of cabbages and other brassicas are produced there each year. Furthermore, 34% of the world's exports of cabbage come from China. India is the world's second-largest producer of cabbage. It harvests about 9,127,000 tons of cabbages annually.[7]

Cabbage is grown on over 38,000 hectares of land in Ethiopia. Combining rainfed and irrigated farming, Ethiopia produces an average of 395,000 tons of cabbage (□□□ □□□) per year. Nevertheless, with an average production of 10.4 tons per hectare, Ethiopia's cabbage productivity is still quite low.[5]

In summary, cabbage is a versatile leafy vegetable with numerous health benefits. It comes in different varieties, is low in calories, and contains essential vitamins and minerals. Cabbage is commonly used in a variety of culinary preparations due to its distinct flavor and texture.

2.2. Cabbage disease

Cabbage, a versatile and nutritious vegetable, can unfortunately be susceptible to various diseases from all of the major classes of pathogens – fungi, water molds, bacteria, viruses, and pests. The infections can range in severity from the unsightly but not usually fatal. Some of these pathogens live in the soil where they can readily attack the roots of crops. Here are some common one

2.2.1. Black rot

Black rot disease affects cabbage worldwide and is caused by the bacterium Xanthomonas campestris. [9] The leaf margins initially turn yellow, and this condition eventually spreads to the leaf's center. A characteristic indication is a golden "V" located in the leaf's midrib. Subsequently, the infection spreads throughout the entire plant, causing the vascular system to turn black. Insects, animals, irrigation, rain, and gardening tools are some of the easy ways that these bacteria might spread in the field. Cabbages can become infected with black rot either mechanically or organically through stomata at the leaf margins.[8] Figure 2.1. shows what Black rot looks like:

.

Figure 1.1 Image of Black Rot

2.2.2. Clubroot

Clubroot is a soil-borne disease caused by the fungus Plasmodiophora brassicae, clubroot leads to swollen, distorted roots and stunted growth.[9] After infecting a crop, may live in the soil for more than 18 years. It can be challenging to identify this long-lived disease. Older plants that are infected will wilt on hot days, but they can often appear to recover after the sun goes down. The pathogen enters the root hairs and then forms large club-like galls that can be as large as five or six inches wide. The roots are exposed to infection by various pathogens that are carried by the soil. Infected seedlings take about three weeks to show signs of root swellings, and they will typically die. If they survive, their slow development may result in significantly lower production. Infections occur when the plants are exposed to spores released from infected roots. Irrigation water, tools, or even footwear can spread these spores.[8] Figure 2.2 shows what Clubroot looks like:

.

Figure 2.2 image of clubroot

2.2.3. Alternaria leaf spot (Dark Spot)

This fungal disease, caused by Alternaria brassicicola, results in circular, dark brown to black spots on leaves.[10] When the temperature is between 68 and 81°F and there is moisture on the cabbage, Alternaria is more prone to spread disease. Yellow dots that enlarge and form rings around them, resembling a target or bull's eye, are among the symptoms. The centers of the tissue may come out as it dies, leaving holes in the leaves. Large patches of dead tissue are formed when the spots converge as the disease progresses.[10]

Controls for Alternaria leaf spot start with good cultural practices. These include rotating your crops, cleaning your gardening tools, using drip irrigation, and getting rid of all dead plant material after the growing season.[10] Figure 2.3 shows what Alternaria leaf spot looks like:

.

Figure 2.3 --Alternaria leaf spot

2.2.4. Downy Mildew

A fungal disease is known as downy mildew caused by the water mold, or oomycete, Peronospora parasitica. Fungal spores appear as white to gray masses on the lower surfaces of the leaves, whereas lesions on the upper surfaces of the leaves are the first signs of this dangerous disease. The vascular system of your seedlings may become black if they are affected. If they have downy mildew, your cabbages could not turn out to be edible.

Raindrops, high humidity, dew, and fog all favor the growth and dissemination of downy mildew. The optimal temperature range for infection is between 46 and 61°F at night and 75°F or lower during the day.

When it comes to fall harvests, downy mildew particular problem in regions with moderate and rainy winters. The pathogen can continue infecting crops throughout the winter, and this can set the stage for severe infections in the spring if the conditions are right.[8] Figure 2.4 shows what Downy Mildew looks like:

.

Figure 2.4-- Downy Mildew

2.2.5. Watery Soft Rot (White Mold)

This bacterial disease, caused by Sclerotinia sclerotiorum, results in water-soaked lesions that turn white and mushy. [10]white mold infects cabbages in the field, and it can also cause losses in storage. In wet weather, sclerotinia can develop between 50 and 77°F. The fungus attacks the stem initially. From there, it can move upward to the leaves and downward to the roots, wilting them and causing the plants to collapse. White, cottony growths on the tissues' surface or inside them are one of the symptoms. This is the reason behind the illness's moniker. The fungus that causes this illness generates reproductive structures known as sclerotia, which contributes to its virulence. These seeds have a dark appearance and have a long shelf life in the ground. These sclerotia can occasionally be seen in the tissue that is affected. Wet conditions favor infection. If the weather turns dry after an infection sets in, you might see brown cankers on the stem that do not progress further.[8] figure 2.5 shows what Watery Rot looks like

.

Figure 2.5-- white mold

2.2.6. Blackroot (Wirestem)

soil-borne fungal disease caused by Rhizoctonia solani, wirestem causes damping-off of seedlings. [9] Contaminated soil can transmit infections to cabbage. Tan or brown lesions on the outer leaves are the first signs. After that, the fungus spreads to the middle of the head, where it might rot totally in ten days. When temperatures are between 68 and 82°F, and the soil is moist, plants are more susceptible to contracting an infection. Once an infection has begun, there are no more controls available.[8] Figure 2.6 shows what the back root looks like:

Figure 2.6-- Image of Black Root

.

2.2.7. Anthracnose

Anthracnose is caused by fusarium oxysporum f. sp. conglutinans, a fungus that lives in the soil. Its spores have a long shelf life. The only way to prevent this deformity on cabbage is to plant resistant varieties because there isn't a practical management plan for yellows. When this pathogen gets into the bloodstream, it usually kills the host completely. The plants that make it will have lifeless, yellow leaves. These leaves may begin to wither at the base early on. Oftentimes, the survivors are yellow on one side of the plant and have stunted development. This is a warm-weather illness, and lows of 68°F are unlikely to cause symptoms to appear.[8] Figure 2.7 shows what yellows look like:

Figure 2.7 Anthracnose disease

.

2.2.9. Healthy cabbage

Healthy cabbage is not affected by any disease. For a healthy cabbage leaf, look for the following characteristics: a vibrant green color without any discoloration or yellowing. Healthy cabbage leaves should feel crisp and firm to the touch. Undamaged Look for leaves that are free from any visible damage, such as holes, tears, or brown spots. the size and shape of cabbage that are well-formed and have a good size.[11] Figure 2.8 shows what healthy cabbage looks like:

Figure 2.8 Health cabbage

2.3. Artificial Intelligence

AI, or Artificial Intelligence, has become a prominent field of study and research in recent years. It refers to the development of intelligent machines capable of performing tasks that would typically require human intelligence. AI encompasses various subfields, including machine learning, natural language processing, computer vision, and robotics. This literature review aims to provide an overview of what AI is and how it has evolved.

AI can be traced back to the mid-20th century, with the invention of digital computers. Initially, AI originated with the development of digital computers in the middle of the 20th century. AI first concentrated on developing computers that could replicate human intelligence in particular domains. Nevertheless, as technology advancements, the discipline grew to include more complex tasks. AI is being used in many different fields these days, such as healthcare, banking, agriculture, transportation, and entertainment.[12]

.

Machine Learning is a crucial subfield of AI. It entails the development of algorithms that let machines learn from data and come to conclusions or predictions without the need for explicit programming. AI has significantly evolved because of this method, which allows computers to adapt and improve performance over time. Supervised learning, reinforcement learning algorithms, and unsupervised learning are all included in machine learning.[13]

Natural language processing, or NLP, is another vital component of artificial intelligence (AI). Its objective is to enable computers to understand and translate human language. This subfield covers tasks including question-answering systems, sentiment analysis, and language translation. To process textual data, NLP systems rely on methods including information extraction, syntactic analysis, and text mining.[12]

Computer vision is another branch of artificial intelligence that focuses on providing computers with the capacity to understand and interpret visual data. It entails activities like facial recognition, image segmentation, and object recognition. Artificial intelligence (AI) systems can analyze and interpret visual input using computer vision, which enables them to carry out activities like autonomous driving, surveillance, and medical imaging analysis.[14]

The field of autonomous systems has advanced significantly as a result of the fusion of AI and robotics. Artificial intelligence (AI)-capable robots can carry out operations like space exploration, surgery, and assembly line operations. These autonomous decision-making robots use artificial intelligence (AI) algorithms to sense and interact with their surroundings. [15]

In conclusion, artificial intelligence (AI) is the creation of intelligent machines that can perform tasks that need human intelligence. With developments in robotics, computer vision, machine learning, and natural language processing, the discipline has changed significantly. Applications of AI are widespread and have the power to completely transform a variety of sectors and societal norms. However, challenges and ethical considerations need to be tackled for the responsible and beneficial development of AI technologies. Figure 2.9 shows *Different branches of AI*

.

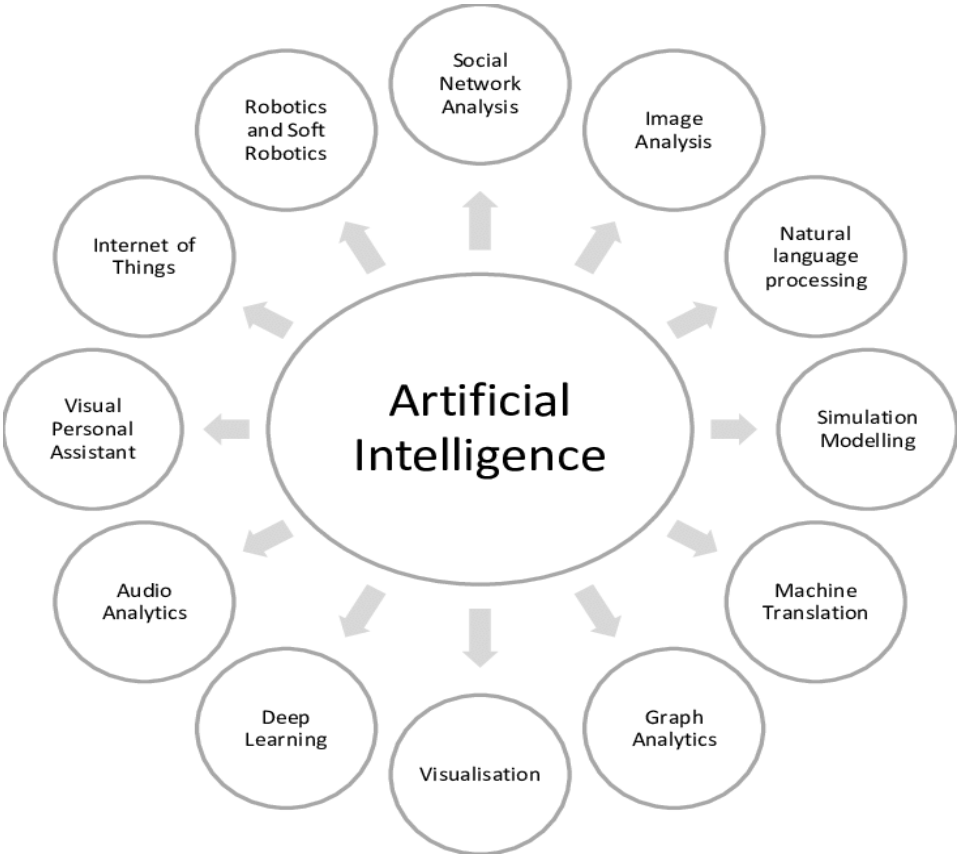


Figure 2.9 Different Branches of AI

2.4. Machin Learning in Agriculture

Machine learning is a rapidly growing discipline and one of the newest technologies being used in information technology to address a variety of issues in fields including engineering, agriculture, medicine, and other aspects of life.

In agricultural operating contexts, large amounts of data technologies and high-performance computers have given rise to machine learning (ML), which has created new opportunities for understanding, quantifying, and comprehending data-intensive processes.[16]

This section discusses a machine learning task. Depending on the learning type (supervised/unsupervised), learning models (classification, regression, clustering, and dimensionality reduction), or the learning models used to accomplish the chosen objective, ML tasks are usually divided into a number of broad categories. Supervised and unsupervised learning

.

are the two main categories into which machine learning falls, depending on the learning signal of the learning system.

The goal of supervised learning is to create a general rule that maps inputs to outputs by presenting example inputs and the related outputs on data. Certain situations may involve partially available inputs that lack some of the desired outputs, or inputs that are only provided as feedback for actions taken in a dynamic environment (reinforcement learning). In the supervised environment, the test data's missing outputs (labels) are predicted using the trained model's acquired expertise. However, with unlabeled data, there is no separation between training and test sets in unsupervised learning. Input data is processed by the learner to find hidden patterns.[16]

According to machine learning's wide range of applications in agriculture, several reviews have been published in this research field. The majority of these review studies have been dedicated to crop disease classification yield prediction, crop recognition, water management, animal welfare, and livestock production.[17]

2.5. Computer Vision in Plant Disease Detection

The artificial intelligence discipline of computer vision aims to give computers the ability to understand visual data, like images or videos, at a high level. It entails creating algorithms and strategies to extract meaningful information from visual inputs, allowing computers to recognize objects, detect patterns, and make decisions based on visual data. Applications for computer vision include driverless cars, medical imaging, image recognition, object detection, and facial recognition.

Computer vision technique for plant leaf disease detection involves a pre-processing stage that consists of image acquisition, color space conversion, image filtering, illumination equalization, and image segmentation discussed.

2.6. Image processing

Image processing is spreading in various fields. One technique that is commonly used for enhancing raw images obtained from different sources is image processing. It is a method for capturing an image and turning it into a digital format so that you can edit it or extract useful information from it. This type of signal distribution uses a picture as the input and produces an image or features associated with the image as the output. The purpose of image processing is

.

distributed into several groups which are given below figure 2.10 shows the image processing technique.

- Visualization: Image processing is used to identify those objects which are not detectable.
- Image sharpening and restoration: To improve an image, a variety of techniques are applied to it during image processing.
- Image retrieval: The user can only detect the relevant section of the image through image processing.
- Pattern measurement: Numerous elements in an image are measured.
- Image Recognition: Substances in an image are recognized. Image processing uses mathematical procedures for the processing of images.[19]

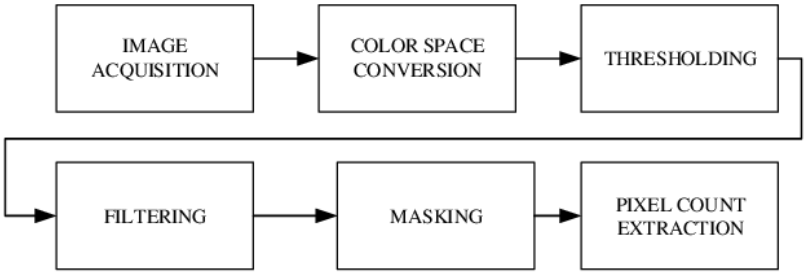


Figure 2.10 Techniques of Image Processing

2.7. Deep learning

Deep learning (DL), a branch of machine learning (ML) and artificial intelligence (AI) is nowadays considered as a core technology of today's Fourth Industrial Revolution. Originating from the artificial neural network (ANN), deep learning (DL) technology gained popularity in the computing community due to its ability to learn from data. These days, it has become popular in many other fields of application, such as visual recognition, text analytics, cybersecurity, healthcare, and agriculture. Hinton et al. introduced "Deep Learning" (DL), which was predicated on the idea of an artificial neural network (ANN). After that, deep learning gained popularity, which led to a revival in neural network research; this is why they are frequently called "new-generation neural networks." This is because deep networks when trained correctly, have demonstrated great performance in a range of classification and detection tasks.[20]

.

Machine learning includes deep learning as one of its subfields. To extract features from the raw data, deep learning employs multiple layers to identify various factors that are important to the input data. Convolutional networks are one type of deep learning technology.

Deep learning is not as efficient compared to traditional machine learning when it comes to handling larger amounts of data. DL technology builds computer models by representing data abstractions using several layers. Compared to other machine learning algorithms, deep learning requires less time to execute during testing, even though it takes a long time to train a model because of its many parameters.

A DL model typically follows the same processing stages as machine learning modeling a three-step processing procedure that includes data analysis and preprocessing, DL model development and training, and validation and interpretation a deep learning workflow for real-world applications. Nevertheless, feature extraction in the DL model is automated as opposed to manual, in contrast to the ML modeling. Machine learning techniques such as k-nearest neighbor, support vector machines, decision trees, random forests, naive Bayes, linear regression, association rules, and k-means clustering are frequently employed in diverse application domains. Conversely, the deep belief network, autoencoder, convolution neural network, recurrent neural network, and numerous more components are part of the DL model.[20] Figure 2.11 shows the deep learning technique.

One prominent type of deep learning is convolutional neural networks (CNNs). CNNs are commonly used for image recognition tasks and have revolutionized computer vision. They consist of different layers, including convolutional layers, pooling layers, and fully connected layers. Convolutional layers perform feature extraction by applying a set of learnable filters to input images, capturing local patterns and spatial hierarchies. Pooling layers then downsample the convolved feature maps, reducing the spatial dimensionality of the extracted features. Finally, fully connected layers take these high-level features and classify them into different classes.

Recurrent neural networks are another kind of deep learning (RNNs). Sequential data, like time series data or spoken language, is what RNNs are meant to process. Recurrent neural networks (RNNs) include a recurrent link, which enables information to be conveyed not only from one layer to the next but also backward to older layers, in contrast to feed-forward neural networks, where information flow is unidirectional. Because of this, RNNs may preserve data from earlier

.

time steps, which makes them useful for tasks like machine translation, speech recognition, and language modeling.[21]

Long Short-Term Memory (LSTM) networks are a popular variant of RNNs that address the vanishing gradient issue, which arises when gradients are too small during backpropagation and impede learning. Three gates—input, forget, and output—as well as an extra memory cell allow LSTMs to regulate how information moves across a network. Because of this, RNNs can circumvent the short-term memory constraints that regular RNNs encounter, which has allowed them to be successfully applied to tasks like speech recognition and language translation.[21]

Another type of deep learning architecture is represented by generative adversarial networks or GANs. The generator and discriminator neural networks make up a GAN. The discriminator network looks for differences between authentic and phony samples, while the generator network creates synthetic data samples. The generator and discriminator networks participate in a competitive learning process in which the generator aims to generate more realistic samples while the discriminator aims to enhance its capacity to identify phony ones. GANs can be utilized for tasks like text generation, image generation, and data augmentation because of this adversarial training, which makes the generator produce samples that are hard to distinguish from genuine data.[22]

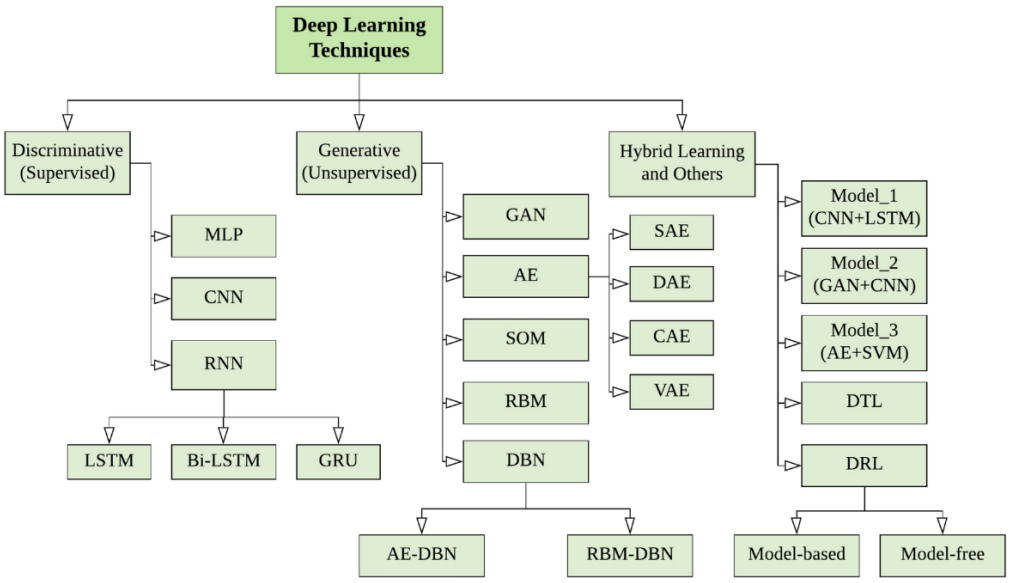


Figure 2.12 deep learning techniques

.

DL techniques are broadly divided into three major categories (i) deep networks for supervised or discriminative learning, (ii) deep networks for unsupervised or generative learning, and (ii) deep networks for hybrid learning and relevant others[20]

Overall, deep learning has proven to be a powerful technique for a wide range of applications, including computer vision, natural language processing, speech recognition, and many others. Through the use of various deep learning types like CNNs, RNNs, LSTMs, and GANs, significant advancements have been made in areas such as image recognition, language translation, and data generation.

2.8. Convolutional Neural Network (CNN)

A convolutional neural network **(CNN)**, also known as ConvNet, stands out as a distinctive type of multilayer neural network specifically tailored for spatial data within the realm of deep learning architectures. The architecture of a CNN is meticulously crafted to autonomously and adaptively grasp the spatial hierarchies of features, spanning from fundamental low-level patterns to more intricate high-level representations. This design is inspired by how living organisms perceive and interact with their surroundings.[23]

CNN is typically composed of three types of layers (or building blocks): convolution, pooling, and fully connected layers. The first two, convolution and pooling layers, perform feature extraction, whereas the third, a fully connected layer, maps the extracted features into final output, such as classification. A convolution layer plays a key role in CNN, which is composed of a stack of mathematical operations, such as convolution, a specialized type of linear operation. In digital images, pixel values are stored in a two-dimensional (2D) grid, i.e., an array of numbers and a small grid of parameters called the kernel, an optimizable feature extractor, at each image position, which makes CNNs highly efficient for image processing, since a feature may occur anywhere in the image. As one layer feeds its output into the next layer, extracted features can hierarchically and progressively become more complex. The process of optimizing parameters such as kernels is called training, which is performed to minimize the difference between outputs and ground truth labels through an optimization algorithm called backpropagation and gradient descent, among others.[24]

.

As we mentioned earlier, a CNN is composed of multiple building blocks (known as layers of the architecture), this subsection, describes some of these building blocks in detail with their role in the CNN architecture.

2.8.1. Building Blocks of CNN Architecture

The CNN architecture includes several building blocks, such as convolution layers, pooling layers, and fully connected layers. A standard architecture consists of one or more fully linked layers stacked on top of a pooling layer and many convolution layers. Forward propagation is the process by which input data are converted into output across these layers.[24] Figure 2.12 shows the building blocks of CNN architecture.

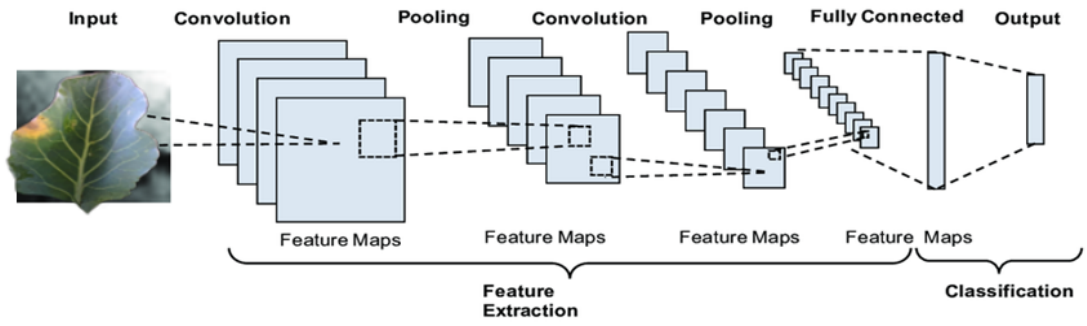


Figure 2.13 Building blocks of CNN architecture

.

2.8.1.1. Convolution Layer

The core element of the CNN architecture is the convolution layer, which carries out feature extraction by forward propagation on a training dataset. Learnable parameters, such as kernels and weights, are then updated by backpropagation using a gradient descent optimization algorithm by the loss value. A rectified linear unit, or ReLU, is made up of an activation function and a convolution operation, two types of linear and nonlinear operations.[24] In addition, the Convolutional layer is the most important component of any CNN architecture. It contains a set of convolutional kernels (also called filters), which get convolved with the input image (N-dimensional metrics) to generate an output feature map.[23]

2.8.1.1.1. Kernel

A kernel can be thought of as a grid of discrete integers or values, where each value represents the kernel's weight. All of a kernel's weights are first assigned random integers when a CNN model is first being trained (there are several methods for initializing the weights as well). After that, the weights are adjusted for every training epoch, and the kernel learns the ability to extract key features.[23] Figure 2.13 shows a 2D kernel

0	1
-1	2

Figure 2.14 Example of a 2 × 2 kernel

.

2.8.1.2. Convolutional Operation

2.8.1.2.1. Padding

The padding is important to give border size information of the input image more importance, otherwise without using any padding the border side features get washed away too quickly. The padding is also used to increase the input image size, as a result, the output feature map size also gets increased.

2.8.1.2.2. Pooling Layer

The feature maps (created after convolution operations) are sub-sampled using the pooling layers; that is, the larger-sized feature maps are shrunk to smaller-sized feature maps. In every pool phase, the most prominent features (or information) are preserved while the feature maps are shrunk. As with the convolution operation, the pooling operation is carried out by defining the operation's stride and the size of the pooled region. There are different types of pooling techniques are used in different pooling layers such as max pooling, min pooling, average pooling, gated pooling, tree pooling, etc**.** [23] Let's describe some of these pooling techniques in detail.

2.8.1.2.2.1 Max Pooling Technique

The max pooling approach is popular in CNNs because it's easy to use and doesn't require any parameter adjusting. A method known as max pooling maximizes the spatial size of a feature map while providing the network translation with translation invariance. The greatest value on the feature map is mostly shown inside a k x k neighborhood to achieve this. The max pooling algorithm finds the largest element in each pooling region. Sparse codes and simple linear classifiers are factors that improve max pooling performance. For the reasons listed above, its popularity has grown in recent years. [25] Figure 2.15 shows the Max pooling operation

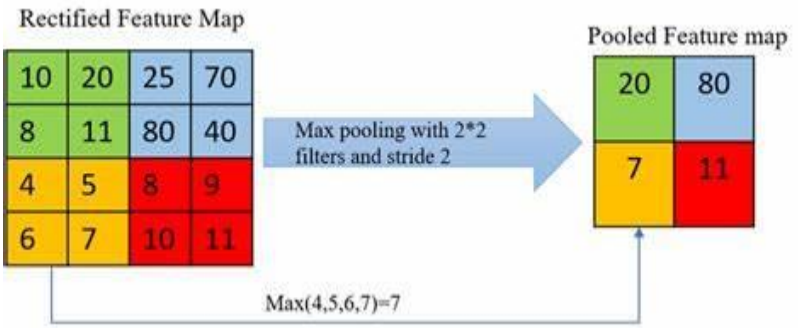


Figure 2.16 Max pooling operation illustration

.

2.8.1.2.2.2 Average Pooling Method

By computing the average values of each of the rectangular pooling areas formed during input segmentation, an average pooling layer down samples the input. To implement the suggested idea, the first convolution-based deep neural network was utilized. An illustration of how the average pooling process operates is shown in Figure 2-16. The image input is split up into multiple distinct rectangular boxes in the average pooling approach. The output channel is displayed once the average of the values in each rectangle's box has been determined.[25]

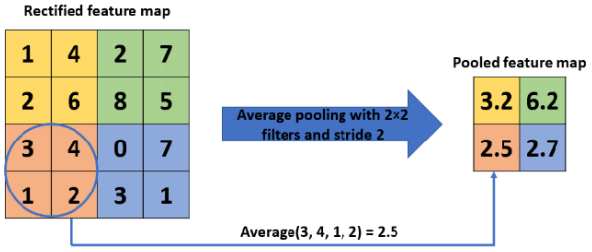


Figure 2.17 Average pooling operation illustration

2.8.2. Activation Functions (Non-linearity)

Any neural network-based model's activation function's primary role is to map the input to the output. To determine the input value, one must first compute the weighted sum of the neuron's input and, if bias exists, add bias to it. In another word, the activation function generates the corresponding output to determine whether or not a neuron fires in response to a given input.[23]

A non-linear activation layer is utilized in CNN design after each learnable layer. The CNN model can learn more difficult tasks and achieve nonlinear input-to-output mapping to those layers' nonlinear behavior. An activation function's ability should be differentiable is crucial for enabling error backpropagation to be used in model training. The most commonly used activation functions in deep neural networks (including CNN) are described below.[23]

.

2.8.2.1 Sigmoid

The sigmoid activation function takes real numbers as its input and binds the output in the range of [0,1]. The curve of the sigmoid function is 'S' shaped.[23]

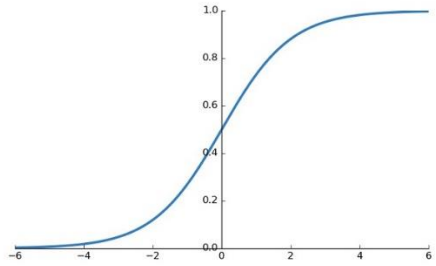


Figure 2.18 Sigmoid

2.8.2.2. **Tanh:**

The Tanh activation function is used to bind the input values (real numbers) within the range of [-1, 1]. The mathematical representation of Tanh is:[23]

$$tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$$

Mathematical formula of the Tanh function

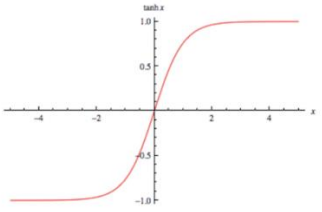


Figure 2.19 Tanh

.

2.8.2.3. **ReLU:**

In Convolutional Neural Networks, the most widely employed activation function is the Rectifier Linear Unit (ReLU). All of the input values are converted to positive numbers using it. ReLU has the advantage of requiring a relatively low computational load in comparison to other methods. ReLU is represented mathematically as: [23]

$f(x)ReLU = max(0, x)$*.....Equation1.1Mathematical Representation of ReLU*

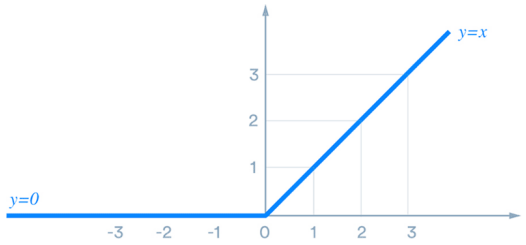


Figure 2.20 ReLU

2.8.2.4. Leaky ReLU

Unlike ReLU, a Leaky ReLU activation function does not ignore the negative inputs completely, rather than down-scaling those negative inputs. Leaky ReLU is used to solve the Dying ReLU problem. The mathematical representation of Leaky ReLU is:[23]

$(x)LeakyReLU = (x, if\ x > 0$..... Equation 2.2 Equation Mathematical Representation of Leaky ReLU

$$\text{mx}, \text{x} \leq 0$$

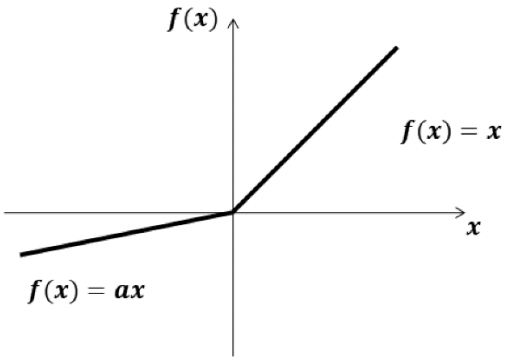


Figure 2.21 Leaky ReLU

.

2.9. Training process of CNN

This section tries to discuss the training or learning process of a CNN model with certain guidelines to reduce the required training time and to improve model accuracy. The training process mainly includes the following steps:

- Data pre-processing and Data augmentation
- Parameter initialization
- Regularization of CNN
- Optimizer selection

Those steps are described in the next subsections

2.9.1. Data pre-processing

When a dataset is artificially transformed to make it cleaner, more featureful, more learnable, and in a consistent format including training, validation, and testing datasets it is referred to as data pre-processing. Before feeding the data into the CNN model, pre-processing the data is completed. It is a known fact that the convolutional neural network (CNN) performance is strongly correlated with the amount of data used to train it. This means that high-quality pre-processing consistently boosts the model's accuracy. Conversely, though, poor pre-processing reduces the model's performance.[26] The general pre-processing techniques that are mostly used are given in the following:

.

2.9.1.1. Mean-subtraction (Zero centering):

Here we substruction the mean from every individual data point (or feature) to make it zero centered as shown in Fig.2.21.

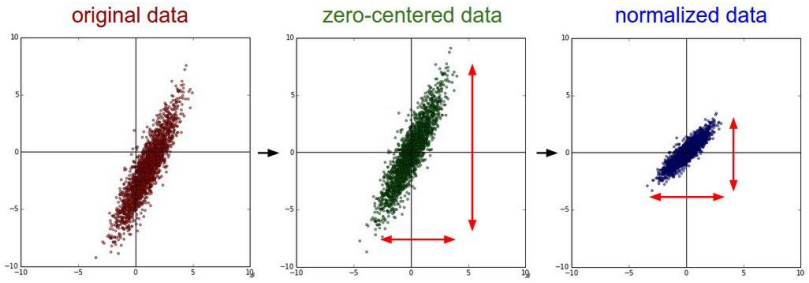


Figure 2.22 Mean-subtraction (Zero centering)

2.9.1.2. Normalization:

Here we normalize the data sample's (belonging from both train, validation, and test datasets) dimension by dividing each dimension by its standard deviations shown in Figure 2.21. The operation would be implemented mathematically as:

$$X_{new} = \frac{X - X_{min}}{X_{max} - X_{min}} \ldots\ldots\ldots\ldots$$ *Equation 3 Mathematical Representation of Normalization*

2.9.2. Data Augmentation

The process of artificially growing or expanding the training dataset is known as data augmentation. Here, we manipulate the data samples (which are exclusive to the training dataset) using various procedures to create one or more new data samples (new versions), which are subsequently employed in the training process. Data augmentation is crucial since, in many cases, just a small training data set is available for the majority of complicated real-world issues, and more training data samples might lead to a CNN model with higher skill levels.[23]

There are several data augmentation operations available such as cropping, rotations, flipping, translations, contrast adjustment, scaling, etc. We can apply those operations separately or in combination to make several new versions from a single data sample. Another reason to use it is that the data augmentation is also able to enforce regularization in the CNN model by avoiding over-fitting problems.[23] Figure 2.23 shows data argumentation

.

Figure 2.23 Data Augmentation

2.9.3. Optimizer selection

we are going to discuss the learning process of the CNN model. The learning process includes two major things, the first one is the selection of the learning algorithm (Optimizer) and the next one is to use several improvements (such as momentum, Adagrad, AdaDelta) to that learning algorithm to improve the result. Any supervised learning algorithm's primary goal is to minimize the error, or loss functions the difference between the expected and actual outputs based on several learnable factors, including weights and biases. Gradient-based learning approaches are a natural choice when learning to a CNN model. To minimize errors, the model iteratively looks for the locally optimal solution throughout each training epoch and updates its parameters continually. The size of parameter updating steps is called the “learning rate” and a complete iteration of parameter update which includes the whole training dataset once is called a “training epoch”. Although the learning rate is a hyper-parameter, we need to choose it so carefully that, it does not affect the learning process badly.

.

2.9.3.1. Gradient Descent or Gradient-Based Learning Algorithm:

To lower the training error, the gradient descent algorithm continually modifies the model parameters throughout each training epoch. To accurately update those parameters, it first determines the objective function's slope (gradient) using a first-order derivative of the model's parameters. Next, to reduce error, it updates the parameter in the gradient's opposite direction. The model's back-propagation procedure, in which the gradient at each neuron is back-propagated to all the neurons belonging to its preceding layer, includes this parameter update process. There exist several variants of the gradient-based learning algorithm, out of them the most widely used are:

- Batch Gradient Descent
- Mini Batch Gradient Descent
- Stochastic Gradient Descent

2.9.3.1.1. Batch Gradient Descent

After the network has processed the entire training dataset, the parameters are only changed once in batch gradient descent. In other words, it uses the gradient to update the parameters after computing the gradient throughout the whole training set. For smaller datasets, the CNN model converges more quickly and generates a more stable gradient when using batch gradient descent. Because the parameters are only changed once for every training epoch, it also consumes fewer resources. However, it takes longer to converge and may converge to the local optimal solution (in the case of non-convex problems) if the training dataset is large.[27]

2.9.3.1.2. Stochastic Gradient Descent (SGD)

In contrast to Batch Gradient Descent, each training sample's parameters are updated separately in this case. Before beginning training, it is advised to randomly shuffle the training sample in each epoch. It is superior to Batch Gradient descent because it converges significantly faster and uses less memory when applied to big training datasets. However, the issue is that it makes extremely noisy progress toward the solution because of the frequent updates, which very unstable the convergence behavior.[28]

.

2.9.3.1.3. Mini Batch Gradient Descent

Here, divide the training instances into many non-overlapping mini-batches, each of which may be thought of as a small sample set. Then, compute the gradients on each mini-batch to update the parameters. Combining Mini Batch Gradient Descent with Stochastic Gradient Descent offers the advantages of both. It also had a consistent convergence and was more memory and computationally-efficient. If the gradient is continually changing during the training process, the weight update changes will be smoothed down by the good momentum factor value. The momentum factor is a hyperparameter.[29]

2.9.3.1.1.1. Momentum

In the objective function of neural networks, momentum is a strategy that adds the gradient computed at the previous training step, weighted by a parameter λ known as the momentum factor. This increases training speed and accuracy. When a problem has a non-convex solution space (or surface), the gradient-based learning technique frequently becomes stuck in a local minimum rather than a global one. This is the main issue with the algorithm.[30]

2.9.3.1.1.2. AdaGrad

The adaptive learning rate method, or AdaGrad, modifies each network parameter differently according to how important it is to solve the problem. In other words, we use a smaller learning rate value for frequent parameters and a larger learning rate value for infrequent parameters. The method involves dividing each parameter's learning rate by the sum of squares of all of the previous gradients for each parameter wij during each training epoch t. AdaGrad is very useful in practice, particularly when dealing with sparse gradients or sparse training data for large-scale neural networks.[30]

2.9.3.1.1.3. Adaptive Moment Estimation

Adaptive Moment Estimation (Adam) is another learning strategy that computes the adaptive learning rate (LR) for every network parameter. It integrates the benefits of Momentum and RMSprop by preserving the exponential moving average of the gradients (similar to Momentum) and the exponential moving average of the squared gradients (similar to RMSprop).[30]

.

2.10. Pre-training of CNN

2.10.1. **LeNet-5**

The LeNet-5 is one of the earliest CNN architectures, which was designed for classifying handwritten digits. It was introduced by LeCun et al. in 1998. The LeNet-5 has 5 weighted (trainable) layers, that is, three convolutional layers and two FC layers. Among them, each of the first two convolution layers is followed by a max-pooling layer (to sub-sample the feature maps) and afterward, the last convolution layer is followed by two fully connected layers. The last layer of those fully connected layers is used as the classifier, which can classify 10 digits.[23] The architecture of LeNet-5 is shown in Figure 2.24.

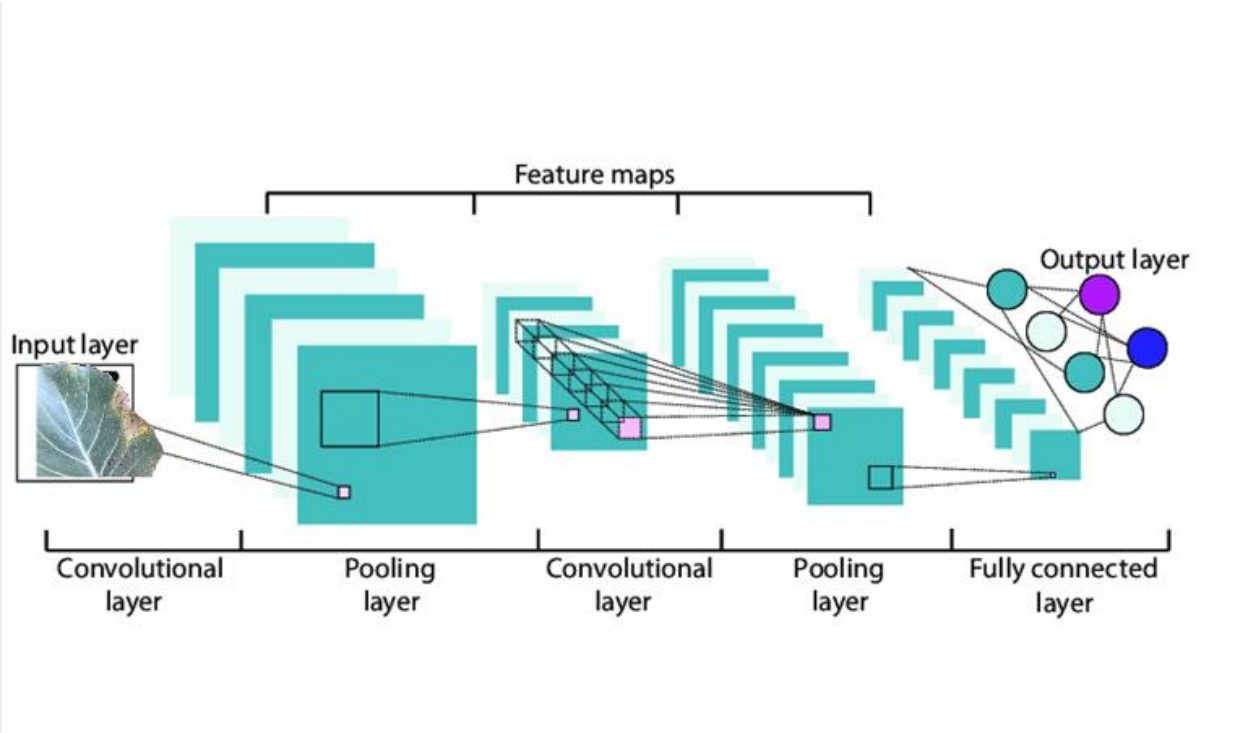


Figure 2.24 The architecture of LeNet-5

.

2.10.2. AlexNet:

Inspired by LeNet, Krizhevky et al. designed the first large-scale CNN model, called AlexNet in 2012, which is designed to classify ImageNet data. It consists of eight weighted (learnable) layers among which the first five layers are convolutional, and afterward, the last three layers are fully connected layers. Since it was designed for ImageNet data, so the last output layer classifies the input images into one of the thousand classes of the ImageNet dataset with the help of 1,000 units. [23]The architecture of AlexNet is shown in Figure: - 2.25.

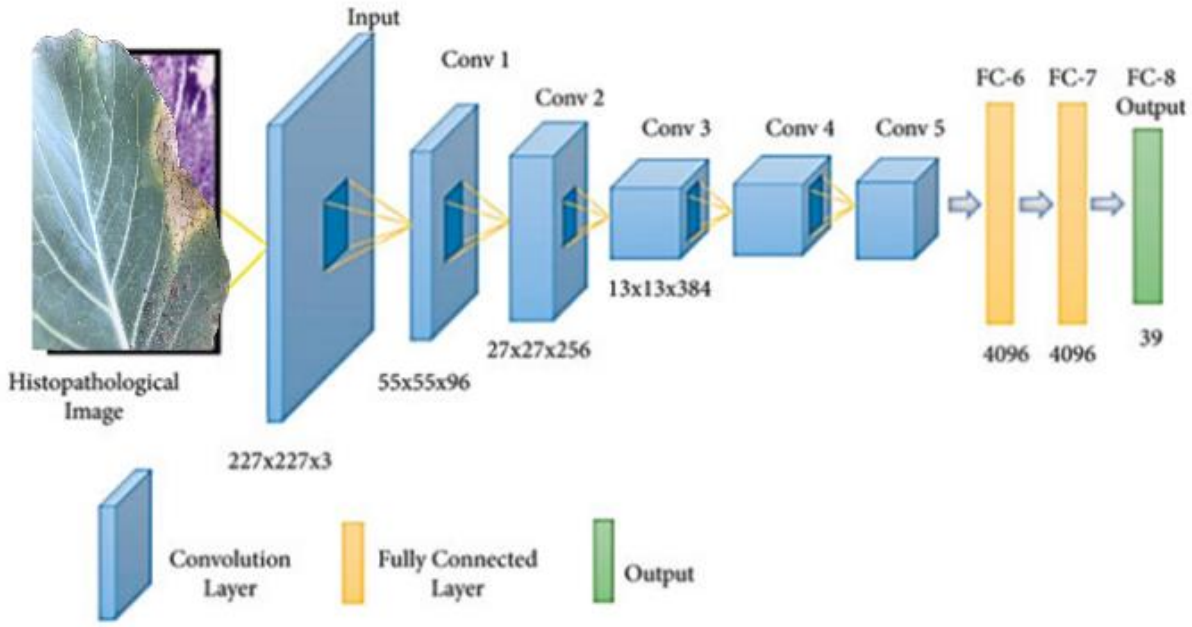


Figure 2.25 The architecture of AlexNet

.

2.10.3. ZFNet

ZFNet was presented by Zeiler and Fergus in ECCV-2014, it has almost similar architecture as AlexNet except that here they used a 7×7 filter with stride 2 in 1'st convolutional layer. In the case of AlexNet, Krizhevky et al. use an 11×11 filter with stride 4 in 1'st convolutional layer. As a result, ZFNet becomes more efficient than AlexNet. [23]The architecture of ZFNet is shown in Fig.2.26

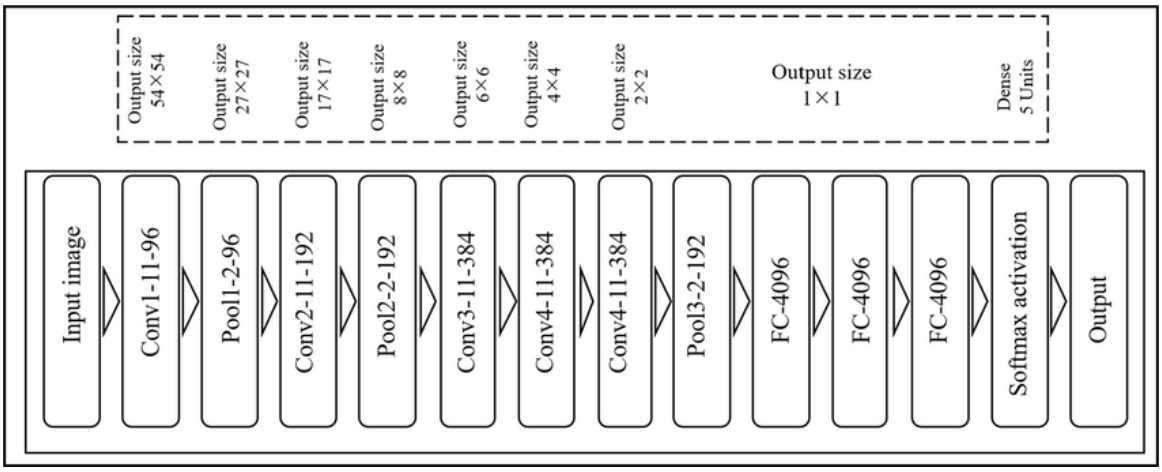


Figure 2.26 The architecture of ZFNet

2.10.4. VGGNet:

VGGNet is one of the most popular CNN architectures, which was introduced by Simonyan and Zisserman in 2014. The authors introduced a total of 6 different CNN configurations, among them the VGGNet-16 (configuration D) and VGGNet-19 (configuration E) are the most successful ones.[23] The architecture of VGGNet-16 is shown in Figure: - 2.27.

.

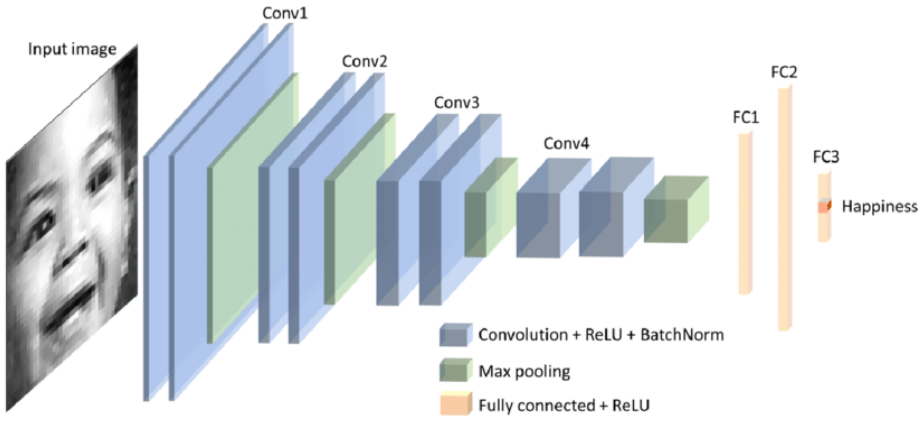


Figure 2.27 The architecture of VGGNet

2.10.5. GoogLeNet:

The GoogleNet architecture is different from all the previously discussed conventional CNN models, it uses network branches instead of using single-line sequential architecture. The GoogleNet was proposed by Szegedy et al. in 2014. The GoogleNet has 22 weighted (learnable) layers, it uses the "Inception Module" as the basic building block of the network. The processing of this module happens in parallel in the network, and each (a simple basic) module consists of 1 × 1, 3 × 3, and 5 × 5 filtered convolution layers in parallel and then it combines their output feature maps, that can result in very high-dimensional feature output. To solve this issue, they used the inception module with dimensionality reduction in their network architecture instead of the naive (basic) version of the inception module.[23]

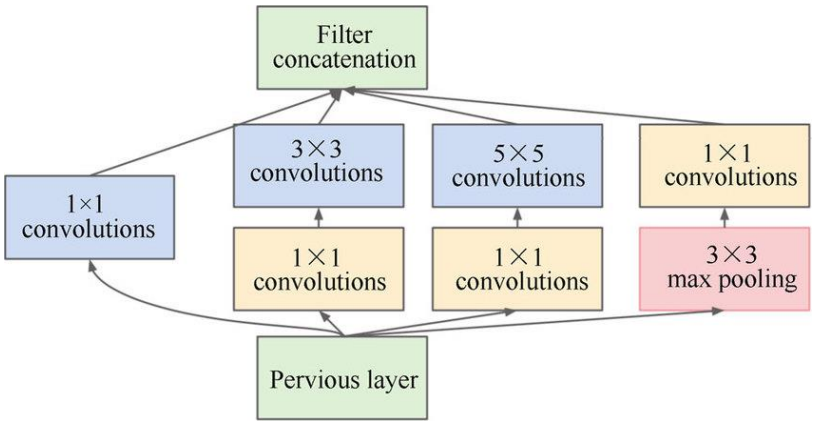


Figure 2.28 Simple Inception Module

.

2.10.6. ResNet:

He et al. from Microsoft, introduced the idea of "identity skip connection" to solve the vanishing gradient problem by proposing the ResNet model. The ResNet's architecture uses residual mapping (H(x) = F(x) + x) instead of learning a direct mapping (H(x) = F(x)) and these blocks are called residual bocks. The complete ResNet architecture consists of many residual bocks with 3 × 3 convolution layers.[23]

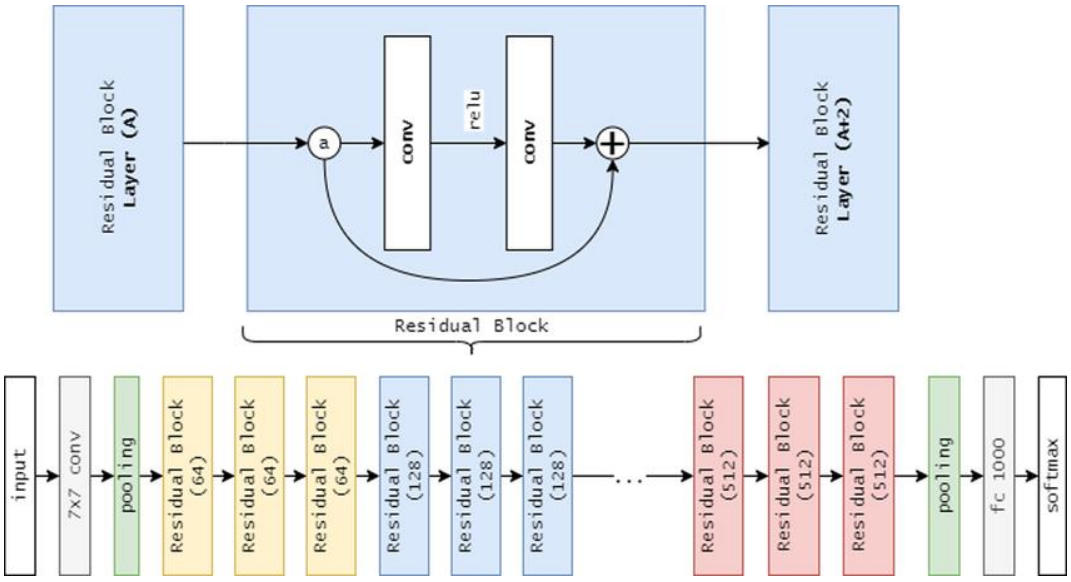


Figure 2.29 The complete ResNet architecture

2.10.7. DenseNet:

DenseNet extends the idea of residual mapping by propagating the output of each block to all subsequent blocks inside each dense block in the network as shown in Figure 2-34. During the model's training process, the information is propagated both forward and backward, strengthening the feature propagation capacity and resolving the vanishing gradient issue. DenseNet was introduced by Huang et al. in 2016 and it became the winner of ILSVRC-2016. Fig.24 shows a DenseNet-based model.[23]

.

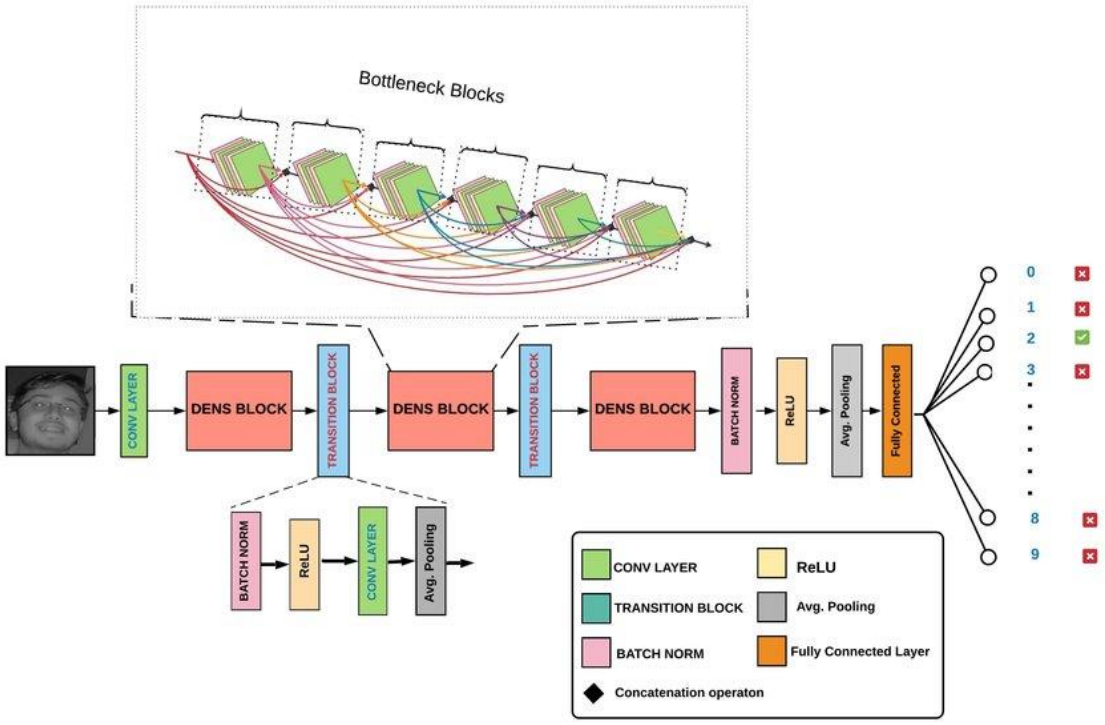


Figure 2.30 A DenseNet-based model with three dense blocks

2.10.8. Inception V3

The inceptionV3 model is an architecture for deep learning CNN from the Inception family that has been widely used for image classification tasks, including plant disease detection. An inception module serves as the main structural component of Inception v3. It incorporates improvements such as label smoothing of convolutions, and an auxiliary classifier to propagate label information lower down the network. Batch normalization is also used for layers in the sidehead. Inception is trained on the ImageNet dataset. Figure 2.25 shows the inception V3 architecture diagram.[31]

.

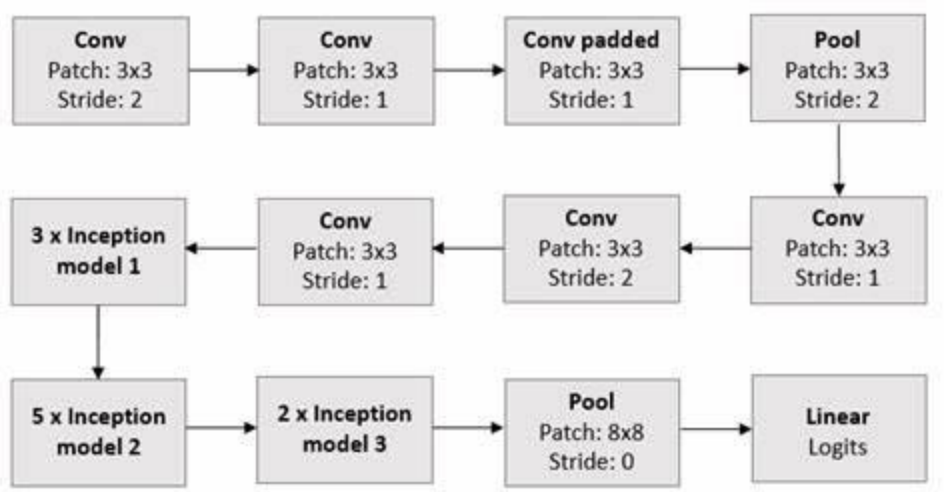


Figure 2.25: - inception V3 architecture diagram

2.11. Hyperparameter Optimization

Machine learning algorithms have been used widely in various applications and areas. A machine learning model needs to have its hyper-parameters adjusted to fit it to various tasks. Machine learning model performance is directly affected by the selection of optimal hyper-parameter setup. Deep knowledge of machine learning algorithms and appropriate hyper-parameter optimization methods is often required. While there are several automated optimization methods available, each one performs differently and has limitations depending on the type of problem. Some important reasons for applying HPO techniques to ML models are as follows [23] and some of the hyperparameters described below

1. It reduces the amount of human effort needed because many machine learning developers take a long time adjusting the hyper-parameters, particularly for big datasets or complex ML algorithms with hyper-parameters.[32]

2. It improves the performance of ML models. Many ML hyper-parameters have different optimums to achieve the best performance in different datasets or problems.[32]

3. It improves the accuracy of the models. Different machine learning algorithms can only be fairly compared when the same level of hyper-parameter tuning process is applied; for this reason, using the same HPO approach to various ML algorithms can also help in identifying the best ML model for a given problem.[32]

.

2.11.1. Hyper-parameter Optimization Techniques

2.11.1.1 Grid search

(GS) is one of the most commonly used methods to explore hyper-parameter configuration space. GS can be considered an exhaustive search or a brute-force method that evaluates all the hyper-parameter combinations given to the grid of configurations. GS works by evaluating the Cartesian product of a user-specified finite set of values. Grid search (GS) is a simple method, its major limitation being that it is time-consuming and impacted by the curse of dimensionality. As a result, it is not appropriate for many hyper-parameters. Furthermore, because GS requires a predetermined, finite set of hyper-parameter values, it frequently fails to identify the global optimum of continuous parameters. Furthermore, with limited time and resources, it is difficult for GS to find integer and continuous hyperparameter optimums. As a result, GS is only effective for a limited number of categorical hyper-parameters when compared to other methods.[32]

2.11.1. 2. Random Search

To overcome certain limitations of GS, random search (RS) was proposed. RS is similar to GS; but, instead of testing all values in the search space, RS randomly selects a pre-defined number of samples between the upper and lower bounds as candidate hyper-parameter values, and then trains these candidates until the defined budget is exhausted. The theoretical basis of RS is that if the configuration space is large enough, then the global optimums, or at least their approximations, can be detected. With a limited budget, RS can explore a larger search space than GS.[32]

The main benefit of RS is its ease of parallelization and resource allocation due to its independent evaluations. As opposed to GS, RS samples a predetermined number of parameter combinations from the given distribution, which decreases the probability of wasting a lot of time on a small, underperforming region and increases system efficiency. Since the number of total evaluations in RS is set to a fixed value of n before the optimization process starts, the computational complexity of RS is O(n). In addition, RS can detect the global optimum or the near-global optimum when given enough budget. [32]

Although RS is more efficient than GS for large search spaces, there are still a large number of unnecessary function evaluations since it does not exploit the previously well-performing regions. To conclude, the main limitation of both RS and GS is that every evaluation in their iterations is

.

independent of previous evaluations; thus, they waste massive time evaluating poorly performing areas of the search space. This issue can be solved by other optimization methods, like Bayesian optimization which uses previous evaluation records to determine the next evaluation.[32]

2.12. Performance Evaluation metric

In this section, we overview different assessment metrics that serve the evaluation of CNN models. For each metric, discuss how it can be computed and the information offered for evaluation.

2.12.1. Accuracy

We discuss different metrics for measuring accuracy for both object classification and detection. Predictions that agree with ground truths are referred to as true positives (TP), predictions that are regarded as inaccurate as false positives (FP), and ground truth annotations that were not identified as false negatives (FN). To calculate the number of TPs in object classifications, one needs just to compare the predicted label with the ground truth label. When it comes to object detection, a bounding box is considered to be TP if the ratio of overlap between the detected box and ground truth, known as the intersection over union (IoU), is above a specific threshold.[33]

2.12.2. Precision

The portion of predictions that are TPs, [33]

$$Precision = \frac{True\ postive}{True\ positive + False\ positive} \ldots\ldots \text{Equation 4 precision}$$

2.12.3. **Recall**

The portion of ground truth annotations that were predicted by the model [33]

$$Recall = \frac{True\ Positive}{True\ positive + False\ negative} \ldots$$

$\ldots\ldots\ldots\ldots\ldots$ *Equation 5 Recall*

.

2.13.4. F1 Score

A metric that combines both precision and recall into a single metric. The F1 score is the harmonic mean of precision and recall. The F1 score provides a way to balance the trade-off between precision and recall, as both measures are important for different aspects of model performance. A high F1 score indicates that a model has both high precision and high recall.[33]

$$F1\ score = \frac{TP}{TP+\frac{1}{2}(FP+FN)} \quad \ldots\ldots\ldots\ldots \textit{Equation 6 F1 score}$$

2.13.5. Mean average precision (mAP)

A metric that takes into account both precision and recall of a model. The metric is calculated by summing the average precision (AP) of each class k over the number of classes K. The AP of each class is calculated by plotting precision for different recall values and calculating the area under the curve (AUC). Compared to the F1 score, mAP is more sensitive to imbalances in the number of training samples between different classes, whilst the F1 score is insensitive to these imbalances in training samples. On the other hand, mAP is more expensive to compute compared to the F1 score.

$$mAP = \frac{1}{N}\sum_{i=1}^{N} APi \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots \textit{Equation 7}$$

Mean average precision

2.13. Related works

This section presents literature related to the area of the proposed study. There are research works that study a general process model to classify a given cabbage leaf image as normal or infected.

The paper presented in [34] deep learning methods to detect diseases in cabbage leaves. The Convolutional Neural Network (CNN) is used to detect cabbage leaf disease, so people with less expertise in software should also use it easily. A total of five hundred images of each type are collected, which are classified into 70% image data for training and 30% image data for testing. The image database is collected from the TensorFlow and Keras libraries. Five major types of diseases, namely leaf miner, diamondback moth, black rot, maggots, and downy mildew, are

.

considered. Sample images of different diseases are collected. Initially, the input images are classified into two types: healthy and diseased. Further, the diseased images are categorized into five different varieties. The main objective of the research work carried out is to recognize and find out whether a cabbage leaf is unhealthy or healthy. Also, it identifies the type of disease that occurred. There are 3000 images; among them, 450 are validation images. All the images are labeled. The proposed model can classify the diseases with 92% accuracy using VGG16. The accuracy of classification can still be increased when trained with a larger dataset and by using pre-trained models.

This paper presents [35]: the detection of black spot disease on kimchi cabbage using hyperspectral imaging and machine learning techniques. The study aims to identify the spectral signature of Alternaria infection on kimchi cabbage and develop an automatic classifier for detecting Alternaria disease symptoms. The dataset collection is based on captured hyperspectral images. They use four classifier models: support vector machine (SVM), random forest (RF), one-dimensional convolutional network (1D-CNN), and one-dimensional residual network (1D-ResNet). The dataset has four classes: healthy, mature symptom, early symptom, and background. The dataset was divided into the training and testing datasets, with 70% for training and 30% for testing. They use a computer equipped with a 16-GB NVIDIA RTX A4000 graphics card and an Intel Core i5 processor. Use a confusion matrix to assess the performance of all models. Overall accuracy is 0.91 for 1D-ResNet, whereas SVM, RF, and 1D-CNN achieved 0.80, 0.88, and 0.86, respectively.

This paper presents [36] the deep learning technology-based night CNN for night crop leaf disease detection. This study aims to identify the disease of plant leaves with great accuracy and efficiency compared to existing approaches. They use a plant village dataset for training the model. Also, the dataset is classified into nine classes of diseases and healthy plant leaves. The night-CNN algorithm is used with a total of 20304 images. The data argumentation process is used to make the number of image sizes balanced for all types of diseases. For evaluation, the images were split into 80% training images and 20% testing images. The disease classification accuracy rate ranges from 93% to 95%.

This paper presents [37] Tomato Leaf Disease Classification via Compact Convolutional Neural Networks with Transfer Learning and Feature Selection. This study proposes a pipeline for the automatic identification of tomato leaf diseases utilizing three compact convolutional neural

.

networks (CNNs). Six classifiers are utilized in the tomato leaf illness identification procedure. These classifiers include Naïve Bayes (NB), K-nearest neighbor (KNN), decision tree (DT), the linear discriminate classifier (LDA), support vector machine (SVM), and quadratic discriminate analysis (QDA). They collected datasets from Plant Village dataset repositories. The dataset is classified into ten classes. Among the classes, one contains sample images of healthy tomato leaves, and the rest of the classes contain various disease-affected tomato leaf image samples. Additionally, the dataset consists of images in several image formats, such as jpg and png. This dataset has been preprocessed by resizing the images, converting them to jpg format, and applying data augmentation techniques. The dataset was pre-split into 2 sections: train data, which contains approximately 80% of the total images, and valid data (considered as test data), which contains approximately 20% of the total images. Executed using MATLAB R2020a and Weka data mining tools. The proposed CNN model has an accuracy of 95.0%.

Paper presented [38] Potato Plant Leaves Disease Detection and Classification using Machine Learning Methodologies. The dataset collected from an openly accessible database of images: 'Plant Village' that consists of diseased as well as healthy images. The obtained dataset consists of 300 potato plant leaves which were categorized into three. Such as the leaves having a disease called Late Blight, the leaves having a disease called Early Blight and the leaves are in a healthy state. The images consist of healthy leaves of about 100 and disease-affected leaves of about 200. The training dataset consists of 70% of the image database i.e., 210 images and the testing dataset consists of the remaining 30% of the image database i.e., 90 images. The K-means algorithm was utilized for the image segmentation, the Gray Level Co-occurrence Matrix was utilized for feature extraction, and multi-class support vector machine methodology along with linear kernel function was utilized for the classification of potato leaves. For evaluation using certain evaluation metrics such as precision, recall, F1-score, and accuracy. overall accuracy of about 95.99%, the precision of about 96.12%, the recall of about 96.25%, and the F1-score of about 96.16%.

.

2.14. Summary of related work

As already stated in related work, there is a study gap in the utilization of deep learning specifically for the classification of cabbage diseases, even though several studies on disease classification in plants using various machine learning approaches have been conducted. Very little research has looked into deep-learning methods for detecting cabbage disease; the majority of papers now in publication concentrate on either specific crop diseases or general plant diseases. There's not enough research on effectively utilizing deep learning for the identification of diseases that affect cabbage.

As observed in a summary of related work, all the papers conducted have some problems that we need to overcome in the thesis. Most of the papers used a dataset from the Plant Village repository. Also, they used single-disease-type identification and a small number of datasets.

Table 11 Summary of related work

Author	Crop type	Objective	Methodology (Approaches)	Result	Research gap
Myna et al. n.d.[33]	cabbage	To detect diseases in cabbage leaves	-Use CNN -image is collected from tensor flow and Keras library.	92% accuracy	Use only a single disease type and there has class imbalance
Kuswidiyanto et. 2023[34]	kimchi cabbage	Identify the spectral signature of Alternaria infection on kimchi cabbage and develop an automatic classifier for detecting	Use SVM, RF,1D-CNN, and 1D-ResNet classifier and use captured hyperspectral images	1D-ResNet, SVM, RF, and 1D-CNN achieved 0.91, 0.80, 0.88, and 0.86	Use only a single disease type, use a limited dataset

.

		Alternaria disease symptoms.		accuracy respectively	
Joshi et al. n.d.[35]	Nightshade Crop	identify the disease of plant leaves with great accuracy and efficiency compared to existing approaches	night-CNN algorithm and use a plant village dataset	accuracy rate ranges from 93% to 95%.	There has class imbalance
Paul et al. n.d.[36]	Tomato	automatic identification of tomato leaf diseases utilizing three compact convolutional neural networks	CNN	95%	-They did not solve cabbage diseases
[37]	Potato	the detection as well as classification of diseases that occur in the potato plants	K-means algorithm	95.99%	-They did not use CNN -use a small number of datasets

.

CHAPTER THREE

3. METHODOLOGY AND PROPOSED ARCHITECTURE

This chapter describes the approaches and methods of data collection used to accomplish this thesis, including methods to implement the model, data collection, data preparation, and software, and hardware configuration of the system used. Additionally, the proposed architecture of the model is briefly described.

3.1. Experimental research methodology

Experimental research methodology is a type of research methodology that uses machine learning to detect and classify cabbage disease using a convolutional neural network (CNN). Experimental research requires both experiments and data collecting to be done. After data collection, we need to train and test their CNN model on a dataset of cabbage. We can plan and carry out experiments employing CNN models to detect and classify cabbage diseases. The CNN-based model's effectiveness will be assessed after it has been trained and tested.

3.2. Research process

The organized and scientific steps researchers take to collect data and assess it to offer the response objective of the research is referred to as the research process. To achieve the objective of this thesis, the following research process is shown in Figure 3.1

.

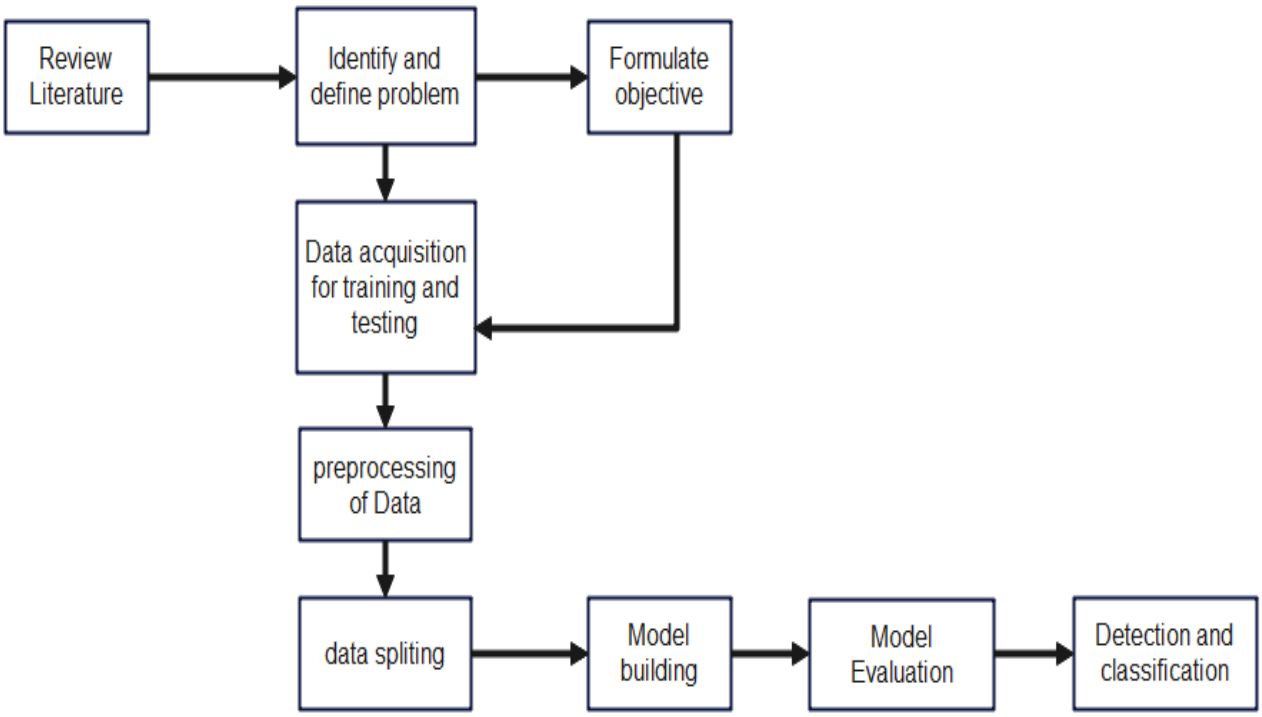


Figure 3.1 Research process

.

3.3. Hardware and software tools

3.3.1. Hardware tools of the model

Laptop computer: - To implement the CNN algorithm with the selected software with HP 250 G7 Notebook PC, Intel Core i7-1065G7 processor, RAM 8 GB, 1 TB hard disk is used.

Camera: - We used iPhone 7 and iPhone 13 phone cameras. The Apple iPhone 7 has a 4.7-inch display with a 750 x 1334 pixels resolution. On the other hand, the Apple iPhone 13 has a 6.1-inch display with an 1170 x 2532 pixels resolution digital camera used to capture the sample images from the field.

External hard disc: - Sufficient storage space to store the dataset, trained models, and any intermediate files generated during development.

3.3.2. Software tools of the model

Several software requirements tools can be used for developing a cabbage disease classification system using the CNN model:

Python programming language: Most deep learning frameworks and libraries are primarily implemented in Python, which makes it the preferred programming language for developing CNN models. So, we used Python language to implement the model.

Google Colab: is a cloud-based service that provides a Jupiter Notebook environment for running code. We used to write and execute Python code directly in the browser, without the need for any local installation for the development of a cabbage disease classification model. With Google Colab, users can collaborate with others, share and access notebooks easily, and take advantage of features like GPU acceleration for deep learning tasks. The service is provided by Google and is free to use.

TensorFlow: is an open-source machine learning framework that provides various tools for building and training CNN models. It offers a wide range of functions and tools for data manipulation, model visualization, and evaluation.

Keras**:** Keras is a high-level neural network API that can run on top of TensorFlow. It provides a user-friendly interface for creating and training CNN models. Keras also includes pre-trained models that can be used as a starting point for cabbage disease classification.

.

OpenCV: OpenCV is an open-source computer vision library that provides tools for image processing and analysis. It can be used for pre-processing the cabbage disease images, such as resizing, noise removal, and segmentation.

Python Imaging Library (PIL): PIL is a library for manipulating images in Python. It provides functions for image loading, cropping, resizing, and basic image processing tasks. PIL can be used for handling the cabbage disease dataset and preparing the images for training.

Mendeley Reference Manager: is a free reference management software that helps our study to organize and manage references and research papers. It allows to import of references from various sources, including online databases and websites, and automatically generates citations and bibliographies in different citation styles.

Microsoft Office Word 2016: we used to write a thesis and draw a table.

Wonder share EdrawMax: is a diagramming tool used to design and draw diagrams of the thesis.

3.4. Proposed architecture of the model

The proposed architecture has different steps. the first step is image data collection acquiring a dataset of cabbage images showing both healthy and different disease symptoms. This dataset is essential for training and evaluating the model. The second step is preprocessing; in preprocessing, the collected images may undergo preprocessing steps such as resizing, normalization, and augmentation to standardize the data and increase the diversity of the samples. The feature extraction step refers to the process of turning row data into a set of relevant, informative, and representative features that can be applied to machine learning. It involves choosing and figuring out features or properties that best represent the fundamental characteristics of the data from the row data. Data augmentation step; The size of the data expands by using these data augmentation techniques, giving the model a more diverse training dataset. These may enhance the model's performance with the unknown dataset. In the model architecture design step, a CNN architecture suitable for the objective of detecting cabbage disease is designed. This includes choosing the right number and type of layers, selecting appropriate activation functions, and organizing the structure of the network. Evaluation step; Using a validation dataset or cross-validation method to evaluate the trained model's performance in the evaluation step. Performance is frequently evaluated using metrics like accuracy, precision, recall, and F1-score. The final step provides outcomes and

.

predicts the health or disease classes of cabbage. The overall model architecture is shown in Figure 3-2.

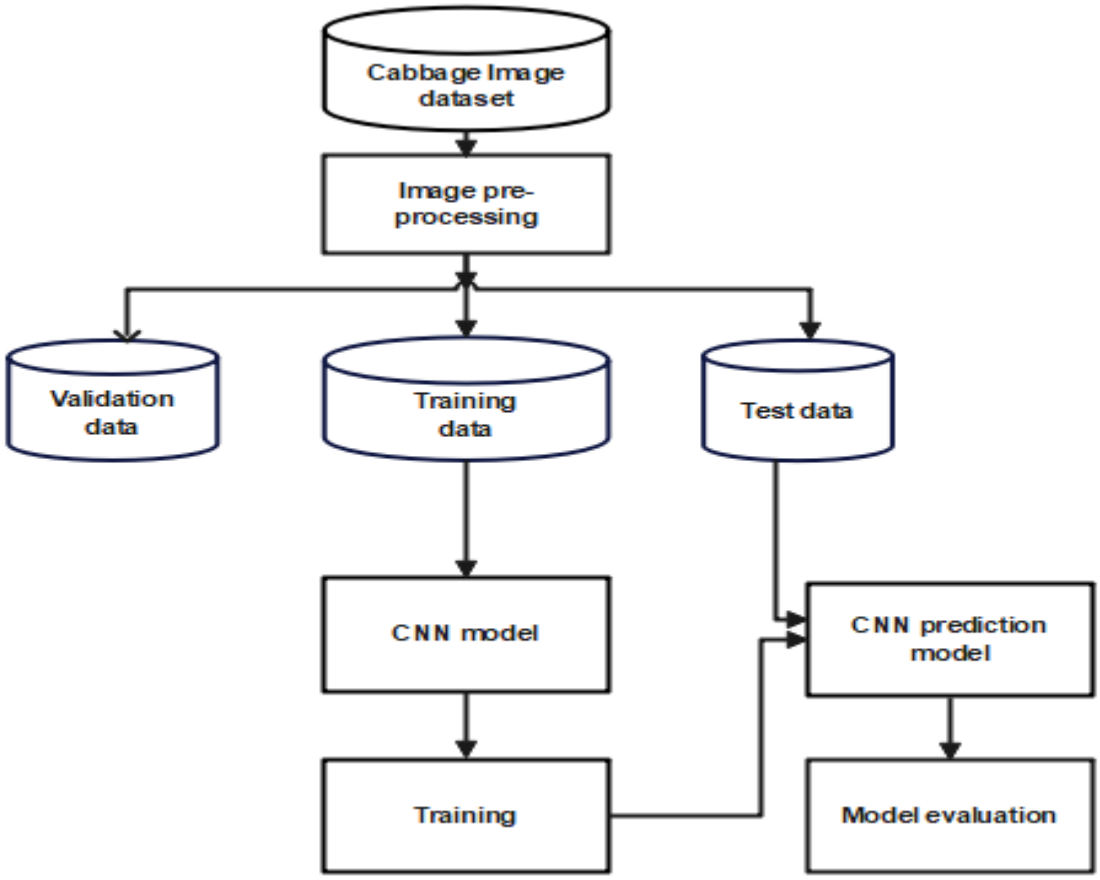


Figure 3.2 Proposed Architecture

3.4.1. Dataset collection and preprocessing

3.4.1.1 Dataset collection

The cabbage leaf image dataset was collected from Sidama region Hawassa zuriya Deneba and East Shewa Zone of the Oromia Region Dugda woreda Meki town cabbage production areas of Ethiopia. 1552 cabbage leaf images were captured by iPhone 7 and iPhone 13 phone cameras. Images captured the growth stage of the cabbage at 40 days,60 days, and 90 days. We have used six classes for classification; namely, Black Rot, Alternaria Leaf Spot disease, Anthracnose, cabbage_worm, Downy_Mildew and Healthy cabbage. The images were classified by Hawassa

.

University Agriculture College Protection Laboratory experts and manually labeled with the relevant classes. The total number of data sets used for experimentation is 5700.

Table 3.1 Total number of data collection

Name of disease and health cabbage	Total Number of sample collections	Number of samples after Augmentations	Total number
Alternaria Leaf Spot	141	809	950
Anthracnose	296	654	950
Black Rot	472	478	950
cabbage_worm	244	706	950
Downy_Mildew	102	848	950
Healthy cabbage	550	400	950
Total number			5700

3.4.1.2 Dataset Augmentation

Data augmentation for cabbage disease classification using convolutional neural networks (CNN) involves artificially expanding the training dataset by applying various transformations to the images of cabbage leaves with different diseases. This prevents overfitting and enhances the model's improvement for generalization. Some common augmentation techniques include image rotation, flipping, scaling, and cropping. These techniques help the model learn a wider range of features and variations in the data, leading to better performance in disease classification tasks

Table 2 Data augmentation parameter

Augmentation parameter	Value
Rotation range	25
Width_shift_range	0.1
Height_shift_range	0.1
Zoom range	0.2
Horizontal flip	True

.

3.4.1.3. Preprocessing of image

Cabbage disease classification using CNN data preprocessing is an essential phase to ensure the input data is in the appropriate format for the model. Data splitting, argumentation, cleaning, and normalization are common steps in the preprocessing phase. These steps are used to reduce complexity and improve the accuracy of a model.

Preprocessing in the case of image data usually involves normalizing the pixel values, resizing the images to a consistent size, and converting images to grayscale. This is done to ensure that pixel values are within a specific range and that the images are the same size. The dimensions of the acquired photographs differ. As a result, we have 1280 * 960 and 960 * 1280 must resize the photos to a uniform 256 by 256 dimension to process them efficiently. The study uses a 256 x 256 image size on the pre-processing part.

```
image = cv2.resize(image, (256,256))
```

`cv2.resize()` is a function in the OpenCV library that resizes an image to a specified size. The first argument of the function is the image to be resized, and the second argument is the desired size of the output image. In this case, the function `cv2.resize(image, (256,256))` resizes the input image to a size of 256x256 pixels.

*Figure 3.3 Original image 1280 *960 pixel*

.

*Figure 3.4 Image of resized 256*256 pixel*

Normalization: normalizing pixel values to a common scale, such as between 0 and 1, helps the model converge faster during training and improves the model's ability to generalize.

Scaling to [0, 1]**:** This method is used to normalize the pixel values of images, which are usually in the range [0, 255]. By dividing each pixel value by 255, we can scale them to the range [0, 1], which is easier for the model to process.

x_train = np. array (x_train, dtype=np. float16) / 225.0

x_test = np. array (x_test, dtype=np. float16) / 225.0

The line of code is a way to convert a list of images (x_train) and (x_test) into a numpy array with a 16-bit floating point data type (dtype=np. float16) and then normalize the pixel values by dividing them by 225.0. This is a preprocessing step for image data before feeding it into a machine-learning model.

By performing these preprocessing steps, the input images are prepared to be fed into the CNN model for training and testing the cabbage disease classification system.

.

3.4.1.4. Feature extraction

To detect and classify cabbage disease using CNN, the feature extraction step involves extracting meaningful and distinguishable features from the input images of cabbage leaves. This is done to illustrate the characteristics unique to different types of cabbage diseases.

Input images are fed into the feature extraction step, and several conventional layers are applied to extract the process through different levels of features. Multiple filters make each convolutional layer convolve over the input image by applying convolution and activation functions.

To reduce their spatial dimensions while preventing the most relevant features, the pooling layers are interspersed convolutionally and are used to down-sample the feature maps. This increases simplicity and protects against overfitting.

Feature maps, once created, pass through fully connected layers. Fully connected layers learn their features by combining features extracted from the previous layer. To produce the final output, the layer performs classification, which represents the probabilities of different cabbage diseases.

The CNN can distinguish between different diseases accurately and identify signification patterns by learning automatically relevant features from the input images. The effectiveness of the cabbage disease classification model feature extraction step is critical, and the model's overall efficiency and performance are improved by this step.

3.4.2. Dataset splitting

We divide the collected dataset into three parts using the train_test_split function from the Sklearn_model_selection module. The training dataset contains 0.7, or 70%, of the datasets. The test subset contains 0.2, or 20%, of the datasets, and the validation dataset contains 10% of the datasets.

3.4.3. Training and validation process

Using the training dataset, train the CNN model. During the training period, the model learns to extract relevant features from the image and classify them into their respective disease categories. Using optimization techniques, pass batches of images through the model and compute the loss and accuracy during the training process.

.

3.4.4. Used algorithm

In this study, we utilize a deep learning algorithm called Convolutional Neural Network (CNN) for the classification of cabbage diseases. This algorithm relies on the automatic identification of relevant features without any human intervention, making it more efficient and scalable than other neural network models. The use of CNN allows for seamless implementation on large networks, streamlining the process of disease classification. It's also more suitable for image data processing than other algorithms.

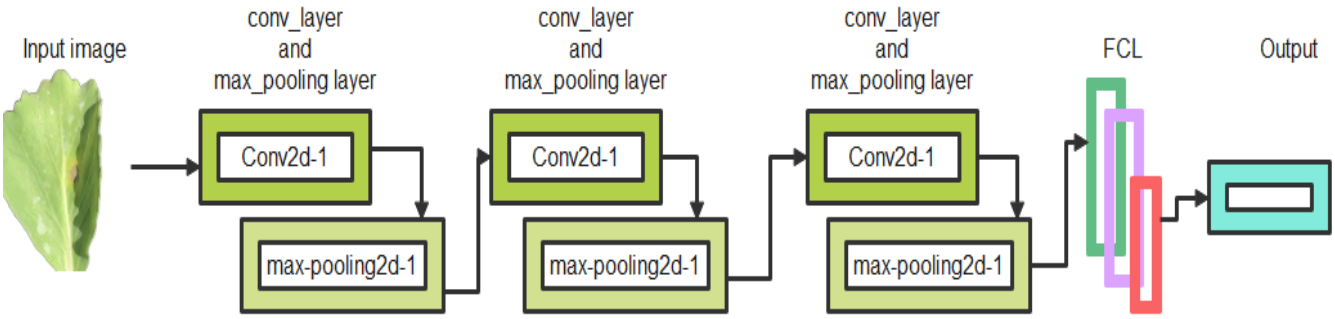


Figure 2.5 CNN architecture

3.4.6. Evaluation metrics

In our cabbage disease classification using CNN, evaluation metrics are essential to assess the performance of the model. These metrics help in measuring the accuracy, precision, recall, and other relevant measures of the model's performance. The study used evaluation metrics:

Accuracy: It is the percentage of correctly classified instances out of the total instances. It provides a general measure of the model's overall performance.[33]

$$Accuracy = \frac{(true\ positives + true\ negatives)}{(total\ predictions)} \quad \ldots equation\ 3.1\ accuracy$$

.

The code: - **metrics = ['accuracy']**

This means we are using the accuracy metric to evaluate the performance of our model. The percentage of correctly identified samples relative to the total number of samples is known as accuracy.

Recall**:** The percentage of true positive predictions among all actual positives, or all disease samples, is often referred to as sensitivity. Recall helps in evaluating the model's capacity to minimize false negatives and detect all positive instances.[33]

$$Recall = \frac{(true\ positives)}{(true\ positives + false\ negatives)} \quad \text{.....Equation 3.2 Recall}$$

```
from sklearn. metrics import recall_score

recall_score(y_true, y_pred)
```

The above code is used to import some functions from the sklearn.metrics module. These functions are used to calculate the recall, which is the ratio of true positives to the total number of actual positive samples. It measures how sensitive the classifier is in detecting the positive class. A high recall score indicates that the classifier has a low false negative rate, meaning that it does not miss many positive samples.

Precision**:** It is the proportion of true positive predictions (correctly identified disease samples) out of all positive predictions (all predicted disease samples). Precision helps in evaluating the model's ability to minimize false positives.[33]

$$Precision = \frac{(true\ positives)}{(true\ positives + false\ positives)} \text{...Equation 3.3 precision}$$

.

from sklearn. metrics import precision_score

`precision_score (y_true, y_pred)`

The above code we used to import some functions from sklearn. metrics module.

`precision_score (y_true, y_pred)` these functions are used to compute the precision, which is the ratio of true positives to the total number of positive predictions It measures the classifier's accuracy in determining the positive class. A high precision score indicates that the classifier has a low false positive rate, meaning that it does not label many negative samples as positive.

Confusion Matrix: The model's predictions compared to the actual values are shown in a tabular form in the confusion matrix. It shows true positives, true negatives, false positives, and false negatives, enabling a detailed analysis of the performance of the cabbage disease classification model.

from sklearn. metrics import confusion matrix

cm = confusion_matrix (Y_true, Y_pred)

The code we used to import the `confusion_matrix` function from the `sklearn. metrics` module. This function calculates the confusion matrix, which is used to assess a classification model's accuracy. Confusion matrix can help us understand how well our model can distinguish between different classes, and identify the sources of errors.

3.4.5. Proposed model description

Input layer: the input layer of our CNN model accepts RGB images of size 256×256×3 with different classes.

Convolutional layer: Conv2D is a convolutional layer that applies a set of filters to the input image and produces a feature map for each filter. The filters are learned during training and can detect various patterns in the image, such as edges, shapes, or textures. The parameters of this layer are the number of filters, the size of the filters, the input shape, the activation function, and the name of the layer. In the proposed model there are three convolutional layers. The first Conv2D layer with 32 filters of size 5x5, an input shape of 256x256x3 (height, width, and channels), a rectified linear unit (ReLU) activation function, and a name of “conv2d_1”. The third Conv2D layer with 32 filters, a kernel size of 3x3, and a ReLU activation function. The fifth Conv2D layer with 64 filters, a kernel size of 3x3, and a ReLU activation function.

.

Pooling layer: MaxPooling2D is a pooling layer that reduces the size of the feature maps by applying a max operation over a sliding window. This helps to reduce the computational cost and avoid overfitting. The parameter of this layer is the size of the window or pool size. In the proposed model there are three MaxPooling2D layers. The second pooling layer with a pool size of 3x3 and a name of "max_pooling2d_1". The fourth and sixth pooling layers with a pool size of 2x2.

Flatten is a layer that reshapes the feature maps into a one-dimensional vector. This is necessary to connect the convolutional layers to the dense layers, which expect a flat input. The parameter of this layer is the name of the layer.

Fully Connected (FC) layer: dense is a fully connected layer that performs a linear transformation on the input vector and applies an activation function. This layer can learn complex patterns and relationships between the features. The parameters of this layer are the number of units, or neurons, and the activation function. In the proposed model, there are two a dense layer with 512 units and a ReLU activation function and a Dense layer with 128 units and a ReLU activation function.

The dropout layer: is a layer that randomly drops out some of the units in the previous layer during training. This helps to prevent overfitting and improve generalization. The parameter of this layer is the dropout rate or the fraction of units to drop. In the proposed model a dropout layer with a dropout rate of 0.25, means that 25% of the units in the previous layer will be randomly set to zero during training.

Output layer: the last layer proposed model is a layer that applies the SoftMax function to the input vector and produces a probability distribution over the output classes. The SoftMax function exponentiates each element and divides the result by the total of the elements to normalize the input vector. This guarantees that the output vector adds up to one and contains values ranging from 0 to 1. The parameter of this layer is the number of output classes. Here is the output of the proposed model summary:

.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_1 (Conv2D)           (None, 252, 252, 32)      2432

 max_pooling2d_1 (MaxPoolin  (None, 84, 84, 32)        0
 g2D)

 conv2d_2 (Conv2D)           (None, 82, 82, 32)        9248

 max_pooling2d_2 (MaxPoolin  (None, 41, 41, 32)        0
 g2D)

 conv2d_3 (Conv2D)           (None, 39, 39, 64)        18496

 max_pooling2d_3 (MaxPoolin  (None, 19, 19, 64)        0
 g2D)

 conv2d_4 (Conv2D)           (None, 17, 17, 64)        36928

 max_pooling2d_4 (MaxPoolin  (None, 8, 8, 64)          0
 g2D)

 flatten_1 (Flatten)         (None, 4096)              0

 dense (Dense)               (None, 512)               2097664

 dropout (Dropout)           (None, 512)               0

 dense_1 (Dense)             (None, 128)               65664

 dense_2 (Dense)             (None, 6)                 774

=================================================================
Total params: 2231206 (8.51 MB)
Trainable params: 2231206 (8.51 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

The model uses a limited number of parameters (2,231,206) compared to other deep learning architectures but performs effectively and efficiently. Other deep learning architectures require a GPU or TPU processor and a large amount of data but the proposed model trends with minimal resources and datasets. When we minimize the CNN architecture layers and parameters, we consume the run time of processes for image classification with available CPU.

.

CHAPTER FOUR

4. RESULT AND DISCUSSION

Experimentation in cabbage disease classification using a convolutional neural network (CNN) involves collecting images of cabbage plants affected by different diseases, as well as healthy cabbage plants. The images are then used to train the CNN model to accurately classify the diseases present in the cabbage plants.

The results of the experimentation would include the accuracy of the CNN model in accurately detecting and classifying the various cabbage diseases. The model's performance can be evaluated based on metrics such as precision, recall, and F1 score.

Overall, the experimentation and results in the cabbage disease classification using CNN aim to provide a reliable and efficient tool for farmers to identify and manage diseases in cabbage plants, ultimately improving crop yield and quality.

4.1. Experimental Environment

For the experimental environment of cabbage disease classification using convolutional neural networks (CNN), the following setup can be used hardware, software, and training process

4.1.1. Hardware and Software

A computer with a high performance for faster training of the CNN models. Python programming language for implementing the CNN models using libraries such as TensorFlow, Keras, and OpenCV. Google Colab notebook IDE for writing and running the code. Image processing libraries for preprocessing the images such as resizing, normalization, and augmentation. A dataset of images of healthy and diseased cabbage plants for training and testing the CNN models has been prepared.

.

4.2. Proposed system model description

A convolutional neural network (CNN) model was implemented using keras as shown in Figer 4.1.

```
model = Sequential()
model.add(Conv2D(32, (5, 5),input_shape = (256,256,3),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, (3, 3),activation='relu',name="conv2d_4"))
model.add(MaxPooling2D(pool_size=(2, 2),name="max_pooling2d_4"))
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(6,activation='softmax'))
model.summary()
```

Figure 4.1 Proposed model

This code defines a convolutional neural network model using the Keras API with the following layers:

The first Conv2D layer has 32 filters with a kernel size of 5x5 and uses a ReLU activation function. The second Conv2D layer has 32 filters with a kernel size of 3x3 and uses a ReLU activation function. The third Conv2D layer has 64 filters with a kernel size of 3x3 and uses a ReLU activation function. The fourth Conv2D layer has 64 filters with a kernel size of 3x3 and uses a ReLU activation function. The first MaxPooling2D layer with a pool size of 3x3 is applied after the first Conv2D layer. The second, third, and fourth MaxPooling2D layer with a pool size of 2x2 is applied after the Conv2D layer. The data is then flattened before being passed to a dense layer with 512 neurons and a ReLU activation function, followed by a dropout layer to reduce overfitting. Another dense layer with 128 neurons and a ReLU activation function is applied. Finally, a dense layer with 6 neurons and a SoftMax activation function is used for classification.

.

4.2.1. Parameter of the layer

In cabbage disease classification using convolutional neural networks (CNN), the parameter layer refers to the specific layer within the neural network architecture where the parameters (i.e., weights and biases) are learned and adjusted during the training process to accurately detect and classify different types of diseases affecting cabbage plants. This layer helps the CNN model to extract and learn features from input images related to cabbage diseases, making it able to make predictions based on the patterns identified in the data. The parameter layer plays a crucial role in the overall performance and accuracy of the disease classification model.

The model consists of four layers input, convolutional, flattened, and fully connected (dense). The input layer reads the input data, while the convolutional layers have different input shapes and filters. The input layer reads input data, resulting in no learnable parameters. Convolutional layers, such as conv2d_1, conv2d_2, conv2d_3, conv2d_4, and conv2d_5, have different input shapes, filters, and parameters. The max_pooling2d_1, max_pooling2d_2 layers, and max_pooling2d_3 layers have no learnable parameters. The flattening layer reshapes the output into a 1D vector, but it also has no learnable parameters. In our experiment, convolutional layers conv2d_1 and conv2d_2 have input shapes (256, 256, 3) and (32, 32, 32), respectively, and 32 filters with parameters of 2432 and 9248, respectively. conv2d_3 and conv2d_4 have input shapes (16, 16, 32) and (8, 8, 64), respectively, and 64 numbers of filters with parameters of 18496 and 36928, respectively. Dense_1, 2, and 3 (output layer) have input sizes of 4096, 512, and 128 with parameters of 2097664, 65664, and 774, respectively. Overall, the total number of learnable parameters in the model is 2,197,706, including weights and biases.

.

4.2.2 Parameters to fit the model

When fitting our CNN model for cabbage disease classification, uses several parameters. These include the number of layers, filter size, activation function, learning rate, batch size, dropout rate, loss function, optimization algorithm, and number of epochs. The CNN architecture includes convolutional, pooling, and fully connected layers. The size of convolutional filters and the step size for sliding filters over input images were also considered. Non-linear activation functions such that ReLU can be used to introduce non-linearity. The learning rate is set to improve performance. The batch size was determined to include the number of samples in each mini-batch for training. Dropout rate was added to prevent overfitting. The loss function is chosen to optimize during training. The optimization algorithm is chosen to update the network's weights based on computed gradients. Number of epochs: Determining the number of complete passes through the training dataset during the training process. These parameters can be adjusted to optimize the performance of our CNN model for cabbage disease classification.

4.2.2.1 Number of layers

The model comprises 12 layers, including Conv2D, MaxPooling2D, Conv2D, Flatten, Dense, Dropout, and Dense layers, each with its unique characteristics and functions.

4.2.2.2 Filter size

Our model has four filter sizes: (5, 5), (3, 3), (3, 3), (3, 3)

4.2.2.3 Activation function

There are a total of 2 activation functions in our model: 'ReLU' activation functions are used in Conv2D layers, Dense layers, and Dropout layers. The 'SoftMax' activation function is used in the final Dense layer.

4.2.2.4 Loss function

The loss function is a measure of how well the model is performing during training. It calculates the difference between the predicted output and the actual output for a given input sample.[39] In our classification use a categorical cross-entropy loss function.

.

4.2.2.6 Learning rate

Below is a table containing performance metrics for a machine learning model at 0.01,0.001 and 0.0001 learning rates. During model development, we compared different learning rates to achieve the best accuracy. The highest accuracy was obtained at a learning rate of 0.001.

Table 4.3 Learning rates

Learning rate	Training			Loss		
	Accuracy	**Validation**	**Test**	**Accuracy**	**Validation**	**Test**
0.01	**0.9269**	**0.8713**	**0.8713**	**0.2436**	**0.3755**	**0.3755**
0.001	**0.9953**	**0.9851**	**0.9851**	**0.0150**	**0.0934**	**0.0934**
0.0001	**0.9310**	**0.8738**	**0.8738**	**0.2993**	**0.3729**	**0.3729**

4.2.2.7 Bach size

Below is a table containing performance metrics for a machine learning model at 32,64 and 128 Bach size. During model development, we compared different Bach size to achieve the best accuracy. The highest accuracy was obtained at a Bach size of 128.

Table 4.4 Bach size

Bach size	Training			Loss		
	Accuracy	**Validation**	**Test**	**Accuracy**	**Validation**	**Test**
32	**0.7281**	**0.7602**	**0.7602**	**1.0168**	**0.8283**	**0.8283**
64	**0.5591**	**0.5794**	**0.5794**	**1.3686**	**1.2484**	**1.2484**
128	**0.9953**	**0.9851**	**0.9851**	**0.0150**	**0.0934**	**0.0934**

.

4.2.2.9 Number of epochs

Below is a table containing performance metrics for a machine learning model at different numbers of epochs. During model development, we compared different numbers of epochs to achieve the best accuracy. The highest accuracy was obtained at epoch 11.

Table 4.5 Number of epochs

Number of epochs	Training			Loss		
	Accuracy	**Validation**	**Test**	**Accuracy**	**Validation**	**Test**
1	**0.6558**	**0.9740**	**0.9740**	**3.3974**	**0.3184**	**0.3184**
11	**0.9953**	**0.9851**	**0.9851**	**0.0150**	**0.0934**	**0.0934**
20	**0.9972**	**0.9703**	**0.9703**	**0.0066**	**0.1293**	**0.1293**
30	**0.9972**	**0.9665**	**0.9665**	**0.0060**	**0.1622**	**0.1622**
40	**0.9963**	**0.9628**	**0.9628**	**0.0047**	**0.1648**	**0.1648**
50	**0.9991**	**0.9740**	**0.9740**	**0.0045**	**0.1846**	**0.1846**
60	**0.9972**	**0.9665**	**0.9665**	**0.0047**	**0.1754**	**0.1754**
70	**0.9981**	**0.9703**	**0.9703**	**0.0053**	**0.1832**	**0.1832**
80	**0.9981**	**0.9703**	**0.9703**	**0.0023**	**0.2018**	**0.2018**
90	**0.9981**	**0.9554**	**0.9554**	**0.0069**	**0.1955**	**0.1955**
100	**0.9981**	**0.9665**	**0.9665**	**0.0019**	**0.2327**	**0.2327**

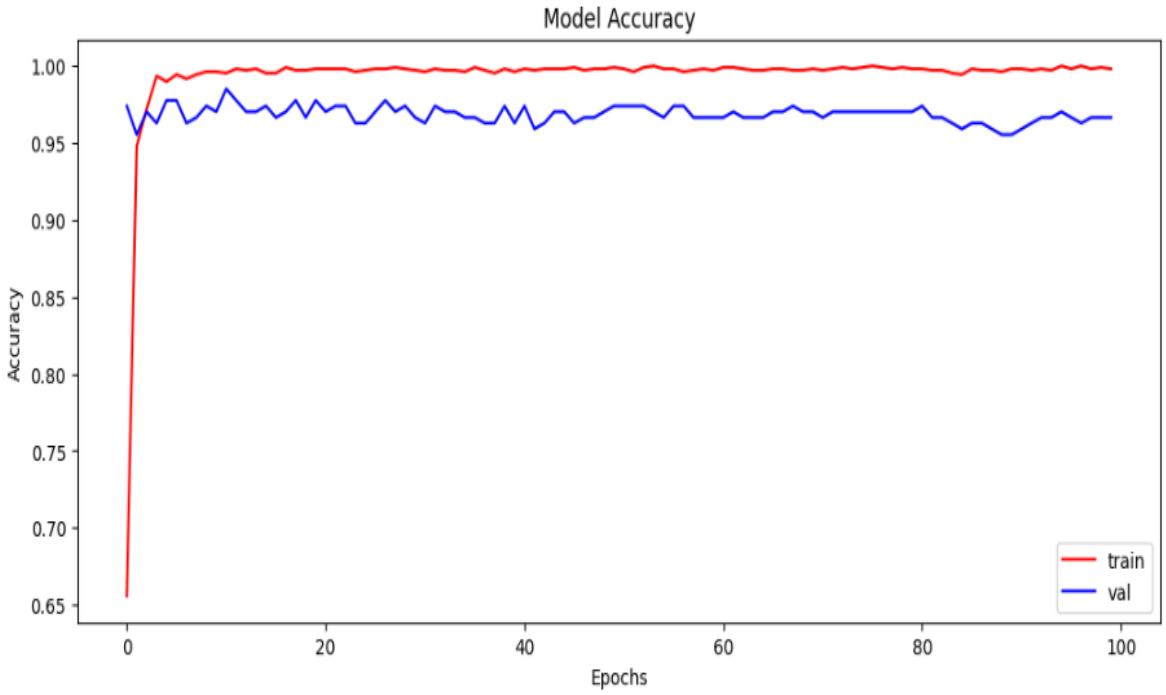
.

4.3. Result Analysis The proposed model

4.3.1. Experimental result of training and validation accuracy curve

The Figure 4.2 displays a line graph titled “Model Accuracy” with two lines representing ‘train’ and ‘Val’. The x-axis is labeled “Epochs” and ranges from 0 to 100, while the y-axis is labeled “Accuracy.” The ‘train’ line remains stable above 99.53% accuracy after an initial sharp increase, and the ‘Val’ line fluctuates but trends around 98.51% accuracy as well. This suggests good generalization of the model being trained.

Figure 4.2 training and validation accuracy curve



`

.

4.3.2. Experimental results of training and validation loss curve

The training and validation losses in Fig.4 are predicted inaccurately. Loss indicates that these models incorrectly classified true data. In other words, the figure 4.3. is a top-to-bottom loss curve. With a given input, the model predicts that the training accuracy loss is 0.0028% and the validation accuracy loss is 0.0078%.

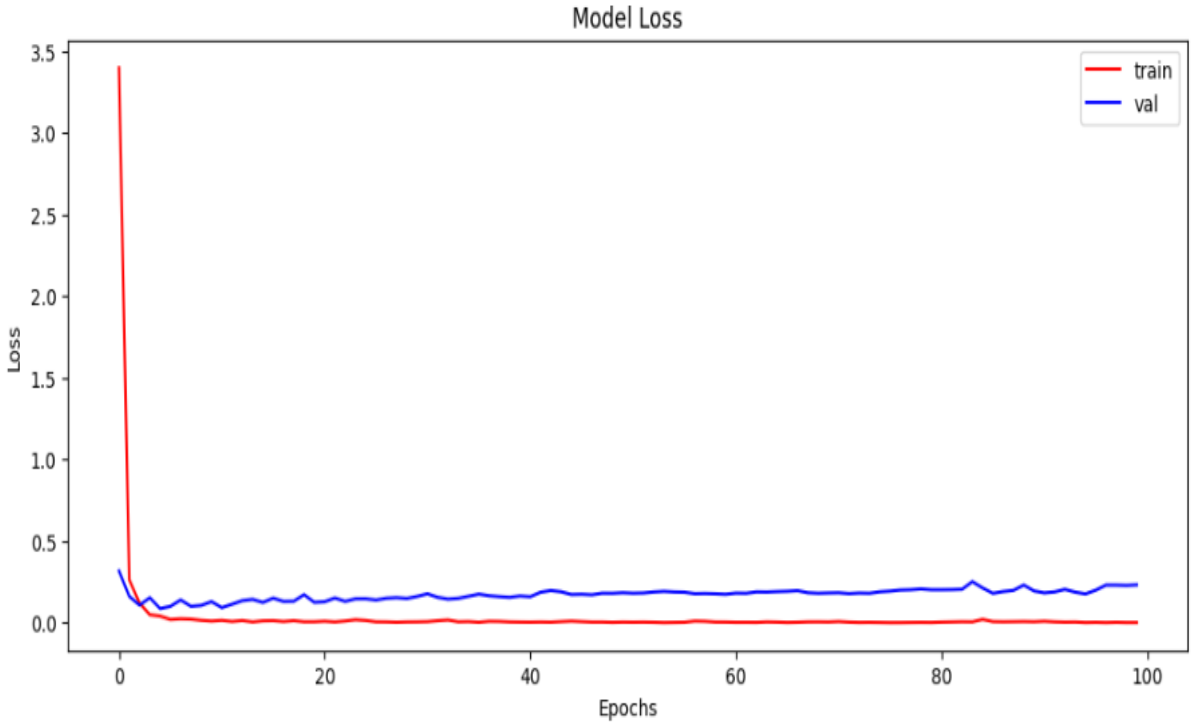


Figure 4.3 Training and validation loss curve

.

4.3.3. Performance of confusion matrix and classification result for the training of the proposed CNN model

The Figure 4.4 shows a confusion matrix, which is a table often used in machine learning to measure the performance of an algorithm. Specifically, it is used for classification problems to visualize the accuracy of a model. The rows represent the actual classes while the columns represent the predicted classes by the model. The classes included are 'Alternaria_leaf_spot', 'Anthracnose', 'Black_rot', 'Downy_Mildew', and 'cabbage_worm'. The diagonal from top left to bottom right shows correct predictions where the predicted class matches the actual class, with values of 40 for Alternaria_leaf_spot, 39 for Anthracnose, 53 for Downy_Mildew, and 46 for cabbage_worm. There are also some misclassifications, as seen by off-diagonal numbers like 1 in the Downy_Mildew row but cabbage_worm column. This matrix is interesting because it provides insight not only into how many predictions were correct but also what types of errors were made by the classification model.

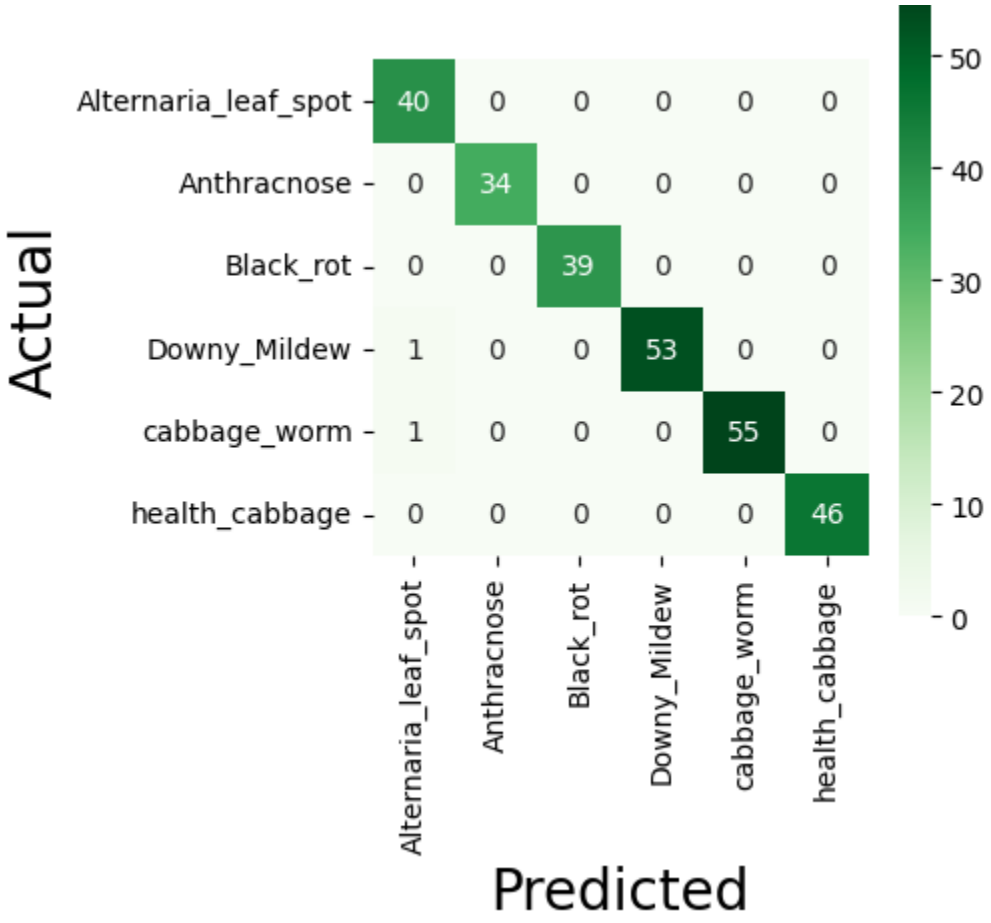


Figure 4.4 Confusion matrix

.

4.3.4. Classification report of the proposed model

A classification report for the proposed model of cabbage disease classification using convolutional neural networks (CNN) would provide detailed insights into the performance of the model. It typically includes metrics such as precision, recall, f1-score, and support for each class. Figure 4.5 shows the classification report of how well the model can correctly identify and classify different types of cabbage diseases.

	precision	recall	f1-score	support
Alternaria_leaf_spot	0.95	0.96	0.95	55
Anthracnose	0.90	0.96	0.93	47
Black_rot	1.00	1.00	1.00	41
Downy_Mildew	1.00	0.92	0.96	50
cabbage_worm	0.97	1.00	0.99	39
health_cabbage	1.00	0.97	0.99	37
accuracy			0.97	269
macro avg	0.97	0.97	0.97	269
weighted avg	0.97	0.97	0.97	269

Figure 4.5. Classification report

In this classification report, we can see the precision of 96.80%, recall of 96.65%, and F1 score of 96.67% for each class, as well as the overall accuracy of the model. These metrics provide valuable information about how well the model is performing in terms of correctly identifying and classifying cabbage diseases.

.

4.3.5. Sample prediction of the proposed model

Figure 4.6 shows the prediction of different disease

Figure 4.6 Sample prediction of different disease

.

4.4 Comparison with Existing Work

The overall performance comparison of our proposed disease classification model with that of some methods is shown in Table 4.5. The comparison results show that our proposed model attains better classification accuracy than the existing methods [33, 34, 35, 36, 37]. When compared to current techniques, it has a high classification accuracy rate. As shown in Table 4.5, the classification accuracy rates of CNN [33, 34, 35, 36], and the K-means algorithm [37] were 92%, 88%, 93%, 95%, and 95.99%, respectively. The comparison results show that the proposed CNN classifier has a better accuracy rate than other classification methods. The proposed system's classification accuracy rate, as shown in Table 4.8, is 98.51%, indicating an average improvement of 2.52% above the best value found in [37].

Table 4.4 Comparison with Existing Work

			Performance measure			
Author	Year Published	Classifier	Accuracy	Precision	Recall	F1-Score
Myna et al. n.d.[33]	2023	CNN	92			
Kuswidiyanto et.[34]	2023	CNN	86			
Joshi et al. n.d.[35]	2022	Night-CNN	95			
Paul et al. n.d.[36]	2023	CNN	95			
[37]	2021	K-means algorithm	95.99	96.12	96.25	96.16
Proposed model		CNN	98.51	96.68	96.65	96.67

.

4.5. Discussion

Our study has yielded promising results on the use of convolutional neural networks (CNN) for the classification of cabbage disease, with an accuracy rate of 98.51%. CNNs are suitable for image classification tasks such as cabbage disease classification because they have shown effectiveness in automatically learning and extracting features from images.

The model achieves a high accuracy rate in classifying cabbage diseases based on images of leaves. The model is trained on a dataset containing images of healthy cabbage leaves and leaves infected with various diseases and accurately classifies the different diseases with a low error rate.

The use of CNNs for cabbage disease classification has the potential to transform the way farmers monitor and manage diseases. By accurately identifying diseases early on, farmers can take proactive measures to prevent the spread of diseases and minimize crop damage.

Overall, the results of using CNNs for cabbage disease classification show great potential for improving crop yield and overall plant health. Further research and development in this area could lead to more advanced and accurate disease classification systems for agriculture.

.

Chapter 5

5. CONCLUSION AND RECOMMENDATION

5.1. Conclusion

Cabbage is important to ensure food security all over the world. In many developing countries, including Ethiopia, the cabbage plant is affected by fungi, viruses, and pests. Currently, the most prevalent cabbage diseases are Alternaria leaf spot, anthracnose, black rot, cabbage worm, downy mildew, and healthy cabbage. The symptoms that appear in parts of the leaves result in reduced production. Cabbage disease can be identified conventionally by the farmer's or an expert's visual inspection. In our study, we used convolutional neural networks for the disease classification of cabbage plants and addressed six types of diseases of cabbage leaves.

Our proposed model addresses the thesis's research questions by collecting 5700 datasets of images of healthy and diseased cabbage plants. Preprocess the images by resizing them to 200 x 200 size, normalizing the pixel values, and augmenting the dataset with techniques like rotation, flipping, and brightness adjustment to increase the variety of images. Design a CNN model architecture that includes multiple convolutional layers followed by pooling layers and fully connected layers. Split the dataset into 70% training and 30% validation and testing sets and train the CNN model using the training set. Experiment with different hyperparameters, such as a learning rate of 0.001 and a batch size of 128 and use the Nadam optimizer to optimize the training process and improve the model's accuracy. Evaluate the model's performance using the validation set and metrics such as 99.53% accuracy, 96.80% precision, 96.65% recall, and a 96.67% F1 score.

Finally, the proposed model yielded 99.53% training accuracy, 98.51% validation accuracy, and 98.51% testing accuracy in classified results from experiments. The outcomes show that the proposed model performs better for classified cabbage leaves.

.

5.2. Contribution

This work contributes to the design and development of a CNN-based model for the classification of cabbage diseases (Alternaria leaf spot, anthracnose, black rot, cabbage worm, downy mildew, and healthy cabbage). Another main contribution of this thesis is the classification of cabbage disease.

5.3. Recommendation for the future

- This study exclusively addressed six label-class diseases of cabbage leaves, but many other diseases affect cabbage not just in the leaves but also in the stems and roots, therefore, it is recommended that future research look into the other diseases.
- Develop a user-friendly software tool for farmers and agricultural experts to easily use the CNN model for cabbage disease classification.

.

Reference

[1] A Systematic Literature Review on Plant Disease Detection: Motivations, Classification Techniques, Datasets, Challenges, and Future Trends," Andreea D, Ioana M. Accessed: May 07, 2024. [Online]. Available: https://www.researchgate.net/publication/371755394_A_Systematic_Literature_Review_on_Plant_Disease_Detection_Motivations_Classification_Techniques_Datasets_Challenges_and_Future_Trends

[2] Y. Gelaye, E. T.-T. S. W. Journal, and undefined 2022, "Agronomic Productivity and Organic Fertilizer Rates on Growth and Yield Performance of Cabbage (Brassica oleracea var. capitata L.) in Northwestern," *hindawi.comY Gelaye, E TadeleThe Scientific World Journal, 2022•hindawi.com*, Accessed: Nov. 27, 2023. [Online]. Available: https://www.hindawi.com/journals/tswj/2022/2108401/

[3] "Cabbages in Ethiopia | The Observatory of Economic Complexity." Accessed: Nov. 28, 2023. [Online]. Available: https://oec.world/en/profile/bilateral-product/cabbages/reporter/eth

[4] T. Getachew, A. Gizachew, G. Fekadu, F. Demis, A. Yenenesh, and T. T. Fasil, "Effects of spacing on yield and head characteristics of cabbage (Brassica oleracea var. capitata L.) in two agro-ecologies of Ethiopia," *Afr J Agric Res*, vol. 18, no. 5, pp. 322–329, May 2022, doi: 10.5897/AJAR2022.15993.

[5] M. S. Thesis, D. K. Gebremeskel, and B. Dar, "M.Sc. Program in Horticulture ASSESSMENT OF PRODUCTION PRACTICES AND EFFECT OF N:P2O5:S RATES ON YIELD AND YIELD COMPONENTS OF HEAD CABBAGE (Brassica Oleracea var. capitata) UNDER IRRIGATION CONDITIONS IN LAY ARMACHIO DISTRICT, AMHARA REGION, ETHIOPA," 2016.

[6] J. Červenski, S. Vlajić, M. Ignjatov, G. Tamindžić, and S. Zec, "Agroclimatic conditions for cabbage production," *Ratarstvo i Povrtarstvo*, vol. 59, no. 2. Institute of Field and Vegetable Crops, pp. 43–50, 2022. doi: 10.5937/ratpov59-36772.

[7] P. Sarkar and H. Raheman, "A Comprehensive Review of Mechanized Cabbage Harvesting Systems and Its Present Status in India," *Journal of The Institution of Engineers (India): Series A*, vol. 102, no. 3, pp. 861–869, Sep. 2021, doi: 10.1007/S40030-021-00557-6/METRICS.

[8] HELGA G EORGE, "Identify, Prevent, and Treat Common Cabbage Diseases | Gardener's Path." Accessed: Nov. 15, 2023. [Online]. Available: https://gardenerspath.com/how-to/disease-and-pests/common-cabbage-diseases/

.

[9] A. P. Keinath, M. A. Cubeta, and D. B. Langston, “Cabbage Disease Ecology and Management,” *Managing Biological and Ecological Systems*, pp. 153–157, Jul. 2020, doi: 10.1201/9780429346170-17/CABBAGE-DISEASE-ECOLOGY-MANAGEMENT-ANTHONY-KEINATH-MARC-CUBETA-DAVID-LANGSTON.

[10] CABI, “Cabbage disorders CABI PEST AND DISEASE PHOTOGUIDE TO,” UK, 2018. [Online]. Available: www.plantwise.org

[11] “Cabbage (Brassica oleracea L.). Overview of the Health Benefits and Therapeutical Uses.” Accessed: May 08, 2024. [Online]. Available: https://www.researchgate.net/publication/353411136_Cabbage_Brassica_oleracea_L_Overview_of_the_Health_Benefits_and_Therapeutical_Uses

[12] Z. Shao, R. Zhao, S. Yuan, M. Ding, and Y. Wang, “Tracing the evolution of AI in the past decade and forecasting the emerging trends,” *Expert Syst Appl*, vol. 209, p. 118221, Dec. 2022, doi: 10.1016/J.ESWA.2022.118221.

[13] K. Yao and Y. Zheng, “Fundamentals of Machine Learning,” *Springer Series in Optical Sciences*, vol. 241, pp. 77–112, 2023, doi: 10.1007/978-3-031-20473-9_3/COVER.

[14] I. Kotseruba, M. Papagelis, and J. K. Tsotsos, “Industry and Academic Research in Computer Vision,” Jul. 2021, Accessed: Nov. 20, 2023. [Online]. Available: https://arxiv.org/abs/2107.04902v3

[15] M. Soori, B. Arezoo, and R. Dastres, “Artificial intelligence, machine learning and deep learning in advanced robotics, a review,” *Cognitive Robotics*, vol. 3, pp. 54–70, Jan. 2023, doi: 10.1016/J.COGR.2023.04.001.

[16] K. G. Liakos, P. Busato, D. Moshou, S. Pearson, and D. Bochtis, “Machine Learning in Agriculture: A Review,” *Sensors (Basel)*, vol. 18, no. 8, Aug. 2018, doi: 10.3390/S18082674.

[17] L. Benos, A. C. Tagarakis, G. Dolias, R. Berruto, D. Kateris, and D. Bochtis, “Machine Learning in Agriculture: A Comprehensive Updated Review,” *Sensors (Basel)*, vol. 21, no. 11, Jun. 2021, doi: 10.3390/S21113758.

[18] “(PDF) COMPUTER VISION-BASED PLANT DISEASE IDENTIFICATION SYSTEM: A REVIEW.” Accessed: Nov. 21, 2023. [Online]. Available: https://www.researchgate.net/publication/366289876_COMPUTER_VISION-BASED_PLANT_DISEASE_IDENTIFICATION_SYSTEM_A_REVIEW

[19] Neetu Rani, “Image Processing Techniques: A Review,” *Journal on Today’s Ideas - Tomorrow’s Technologies*, vol. 5, no. 1, pp. 40–49, Jun. 2017, doi: 10.15415/JOTITT.2017.51003.

.

[20] I. H. Sarker, “Deep Learning: A Comprehensive Overview on Techniques, Taxonomy, Applications and Research Directions,” *SN Comput Sci*, vol. 2, no. 6, pp. 1–20, Nov. 2021, doi: 10.1007/S42979-021-00815-1/FIGURES/6.

[21] R. DiPietro and G. D. Hager, “Deep learning: RNNs and LSTM,” *Handbook of Medical Image Computing and Computer Assisted Intervention*, pp. 503–519, Jan. 2019, doi: 10.1016/B978-0-12-816176-0.00026-0.

[22] I. Goodfellow *et al.*, “Generative adversarial networks,” *Commun ACM*, vol. 63, no. 11, pp. 139–144, Oct. 2020, doi: 10.1145/3422622.

[23] A. Ghosh, A. Sufian, F. Sultana, A. Chakrabarti, and D. De, “Fundamental concepts of convolutional neural network,” *Intelligent Systems Reference Library*, vol. 172, pp. 519–567, 2019, doi: 10.1007/978-3-030-32644-9_36.

[24] R. Yamashita, M. Nishio, R. K. G. Do, and K. Togashi, “Convolutional neural networks: an overview and application in radiology,” *Insights Imaging*, vol. 9, no. 4, pp. 611–629, Aug. 2018, doi: 10.1007/S13244-018-0639-9/FIGURES/15.

[25] A. Zafar *et al.*, “A Comparison of Pooling Methods for Convolutional Neural Networks,” *Applied Sciences 2022, Vol. 12, Page 8643*, vol. 12, no. 17, p. 8643, Aug. 2022, doi: 10.3390/APP12178643.

[26] K. S. Sudeep and K. K. Pal, “Preprocessing for image classification by convolutional neural networks,” *2016 IEEE International Conference on Recent Trends in Electronics, Information and Communication Technology, RTEICT 2016 - Proceedings*, pp. 1778–1781, Jan. 2017, doi: 10.1109/RTEICT.2016.7808140.

[27] D. R. Wilson and T. R. Martinez, “The general inefficiency of batch training for gradient descent learning,” *Neural Networks*, vol. 16, no. 10, pp. 1429–1451, Dec. 2003, doi: 10.1016/S0893-6080(03)00138-2.

[28] X. Dai and Y. Zhu, “Towards Theoretical Understanding of Large Batch Training in Stochastic Gradient Descent,” Dec. 2018, Accessed: May 08, 2024. [Online]. Available: https://arxiv.org/abs/1812.00542v1

[29] S. Khirirat, H. R. Feyzmahdavian, and M. Johansson, “Mini-batch gradient descent: Faster convergence under data sparsity,” *2017 IEEE 56th Annual Conference on Decision and Control, CDC 2017*, vol. 2018-January, pp. 2880–2887, Jun. 2017, doi: 10.1109/CDC.2017.8264077.

[30] A. Defazio, “Adaptivity without Compromise: A Momentumized, Adaptive, Dual Averaged Gradient Method for Stochastic Optimization,” *Journal of Machine Learning Research*, vol. 23, pp. 1–34, 2022, Accessed: May 08, 2024. [Online]. Available: https://github.com/facebookresearch/madgrad

.

[31] L. Yang and A. Shami, “On hyperparameter optimization of machine learning algorithms: Theory and practice,” *Neurocomputing*, vol. 415, pp. 295–316, Nov. 2020, doi: 10.1016/J.NEUCOM.2020.07.061.

[32] “Confusion Matrix, Accuracy, Precision, Recall, F1 Score | by Harikrishnan N B | Analytics Vidhya | Medium.” Accessed: May 08, 2024. [Online]. Available: https://medium.com/analytics-vidhya/confusion-matrix-accuracy-precision-recall-f1-score-ade299cf63cd

[33] A. Myna, K. Manasvi, J. Pavan, and H. R.- perspective, “Classification and Detection of Cabbage Leaf Diseases from Images Using Deep Learning Methods,” *article.autocis.netAN Myna, K Manasvi, JK Pavan, HS Rakshith, DJ Yukthaperspective•article.autocis.net*, Accessed: Nov. 27, 2023. [Online]. Available: http://article.autocis.net/pdf/10.11648.j.acis.20231101.11.pdf

[34] L. W. Kuswidiyanto, D. E. Kim, T. Fu, K. S. Kim, and X. Han, “Detection of Black Spot Disease on Kimchi Cabbage Using Hyperspectral Imaging and Machine Learning Techniques,” *Agriculture 2023, Vol. 13, Page 2215*, vol. 13, no. 12, p. 2215, Nov. 2023, doi: 10.3390/AGRICULTURE13122215.

[35] B. M. Joshi and H. Bhavsar, “International Journal of INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING Deep Learning Technology based Night-CNN for Nightshade Crop Leaf Disease Detection.” [Online]. Available: www.ijisae.org

[36] S. Paul, A. Biswas, A. Saha, M. Zulfiker, N. R.- Array, and undefined 2023, “A real-time application-based convolutional neural network approach for tomato leaf disease classification,” *Elsevier*, Accessed: Nov. 27, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2590005623000383

[37] S. D. Allen *et al.*, “Potato plant leaves disease detection and classification using machine learning methodologies,” *iopscience.iop.orgA Singh, H KaurIOP Conference Series: Materials Science and Engineering, 2021•iopscience.iop.org*, doi: 10.1088/1757-899X/1022/1/012121.

[38] Y. Tian, D. Su, S. Lauria, and X. Liu, “Recent advances on loss functions in deep learning for computer vision,” *Neurocomputing*, vol. 497, pp. 129–158, Aug. 2022, doi: 10.1016/J.NEUCOM.2022.04.127.

.

APPENDIX

Appendix I: Importing Library

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib.image import imread
import cv2
import random
import os
from os import listdir
from PIL import Image
from sklearn.preprocessing import label_binarize, LabelBinarizer
from keras.preprocessing import image
from keras_preprocessing.image import img_to_array, array_to_img
from keras.optimizers import Adam
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D
from keras.layers import Activation, Flatten, Dropout, Dense
from sklearn.model_selection import train_test_split
from keras.models import model_from_json
from keras.utils import to_categorical
```

Appendix II: Preprocessing

```
for filename in os.listdir(path):
    if filename.endswith('.jpg'):
        with Image.open(os.path.join(path, filename)) as img:
            img_resized = img.convert('RGB').resize((200, 200))
            img_resized.save(os.path.join('/content/drive/MyDrive/health_cabbage', filename))
```

Appendix III: Load dataset

```
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 ="/content/drive/MyDrive/DATAset/Datasetsamri/train"
test_dir="/content/drive/MyDrive/DATAset/Datasetsamri/test"
```

.

Appendix IV: Data Augmentation

```
Generates batches of image data with data augmentation
datagen = ImageDataGenerator(rotation_range= 25, # Degree range for random rotations
                         width_shift_range=0.1, # Range for random horizontal shifts
                         height_shift_range=0.1, # Range for random vertical shifts
                         zoom_range=0.2, # Range for random zoom
                         horizontal_flip=True, # Randomly flip inputs horizontally
```

Appendix V: Data Split

```
x_train, x_test, y_train, y_test = train_test_split(image_list, label_list, test_size=0.2, random_state = 10)
```

Appendix VI: Model Building

```
model = Sequential()
model.add(Conv2D(32, (5, 5),input_shape = (256,256,3),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, (3, 3),activation='relu',name="conv2d_4"))
model.add(MaxPooling2D(pool_size=(2, 2),name="max_pooling2d_4"))
model.add(Flatten(name="flatten_1"))
model.add(Dense(512,activation='relu'))
model.add(Dropout(0.15))
model.add(Dense(128,activation='relu'))
model.add(Dense(6,activation='softmax'))
model.summary()
```

.

Appendix VII: Train Model

```
from tensorflow.keras.optimizers import Nadam

model.compile(loss = 'categorical_crossentropy',
              optimizer = Nadam(0.001),
              metrics = ['accuracy'])
```

```
epochs =100
batch_size = 128
history = model.fit(x_train, y_train,
                    batch_size = batch_size,
                    epochs = epochs,
                    validation_data = (x_val, y_val))
```

```
Epoch 1/100
9/9 [==============================] - 92s 9s/step - loss: 3.3974 - accuracy: 0.6558 - val_loss: 0.3184 - val_accuracy: 0.9740
Epoch 2/100
9/9 [==============================] - 91s 10s/step - loss: 0.2640 - accuracy: 0.9479 - val_loss: 0.1627 - val_accuracy: 0.9554
Epoch 3/100
9/9 [==============================] - 86s 9s/step - loss: 0.1207 - accuracy: 0.9721 - val_loss: 0.1084 - val_accuracy: 0.9703
Epoch 4/100
9/9 [==============================] - 94s 10s/step - loss: 0.0504 - accuracy: 0.9935 - val_loss: 0.1539 - val_accuracy: 0.9628
Epoch 5/100
9/9 [==============================] - 90s 10s/step - loss: 0.0417 - accuracy: 0.9898 - val_loss: 0.0877 - val_accuracy: 0.9777
Epoch 6/100
9/9 [==============================] - 91s 10s/step - loss: 0.0209 - accuracy: 0.9944 - val_loss: 0.1009 - val_accuracy: 0.9777
Epoch 7/100
9/9 [==============================] - 99s 11s/step - loss: 0.0248 - accuracy: 0.9916 - val_loss: 0.1406 - val_accuracy: 0.9628
Epoch 8/100
9/9 [==============================] - 90s 10s/step - loss: 0.0228 - accuracy: 0.9944 - val_loss: 0.1013 - val_accuracy: 0.9665
Epoch 9/100
9/9 [==============================] - 89s 10s/step - loss: 0.0153 - accuracy: 0.9963 - val_loss: 0.1068 - val_accuracy: 0.9740
Epoch 10/100
9/9 [==============================] - 91s 10s/step - loss: 0.0109 - accuracy: 0.9963 - val_loss: 0.1302 - val_accuracy: 0.9703
Epoch 11/100
9/9 [==============================] - 86s 9s/step - loss: 0.0150 - accuracy: 0.9953 - val_loss: 0.0934 - val_accuracy: 0.9851
Epoch 12/100
9/9 [==============================] - 91s 10s/step - loss: 0.0084 - accuracy: 0.9981 - val_loss: 0.1141 - val_accuracy: 0.9777
Epoch 13/100
9/9 [==============================] - 92s 10s/step - loss: 0.0140 - accuracy: 0.9972 - val_loss: 0.1363 - val_accuracy: 0.9703
Epoch 14/100
9/9 [==============================] - 95s 11s/step - loss: 0.0056 - accuracy: 0.9981 - val_loss: 0.1439 - val_accuracy: 0.9703
Epoch 15/100
9/9 [==============================] - 92s 10s/step - loss: 0.0121 - accuracy: 0.9953 - val_loss: 0.1251 - val_accuracy: 0.9740
Epoch 16/100
```

.

Appendix VIII: Plot the training History

```
plt.figure(figsize = (12,5))
plt.plot(history.history['accuracy'], color = 'r')
plt.plot(history.history['val_accuracy'], color = 'b')
plt.title('Model Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend(['train', 'val'])
plt.show()
```

Appendix IX: Evaluation metrics

```
Y_pred = model.predict(x_val)

Y_pred = np.argmax(Y_pred, axis=1)
Y_true = np.argmax(y_val, axis=1)
cm = sklearn.metrics.confusion_matrix(Y_true, Y_pred)
plt.figure(figsize=(3, 3))
ax = sns.heatmap(cm, cmap=plt.cm.Greens, annot=True, square=True, xticklabels=all_labels, yticklabels=all_labels)
ax.set_ylabel('Actual', fontsize=20)
ax.set_xlabel('Predicted', fontsize=20)
```