الجمهورية الجزَائِرية الديمقرَاطية الشعبية
PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA
وزَارة التعليم العَالي و البحث العلمي
MINISTRY OF HIGHER EDUCATION AND SCIENTIFIC RESEARCH
جَامعة عمَار ثليجي بِالأَغوَاط
UNIVERSITY OF LAGHOUAT

FACULTY OF SCIENCES
DEPARTMENT OF COMPUTER SCIENCE

Field : Mathematics and Computer Sciences
Option : Computer Science
Specialization : Information Systems and Decision-Making.

Memoire Submitted in Partial fulfillment of the requirements for the Master degree in Computer Science

Submitted by: Abdelatif Hamdi

THEME

Wheat Leaf Diseases Classification using Deep Learning Techniques

Jury members:

Mr	Mohamed Ridha Bouzidi	MC(B)	(University of Laghouat)	President
Mr	Lakhdar Kamal Ouladdjedid	MC(B)	(University of Laghouat)	Examiner
Mr	Tahar Bendouma	MC(A)	(University of Laghouat)	Examiner
Mr	Younes Guellouma	MC(A)	(University of Laghouat)	Advisor

2023

الجمهورية الجزَائِرية الديمقرَاطية الشعبية
République Algérienne Démocratique et Populaire
وزَارة التعليم العَالي و البحث العلمي
Ministère de l 'Enseignement Supérieur et de la Recherche Scientifique
جَامعة عمَار ثليجي بِالأَغوَاط
UNIVERSITÉ AMAR TELIDJI LAGHOUAT

FACULTE DES SCIENCES
DEPARTEMENT D'INFORMATIQUE

Domaine : Mathématiques et Informatique
Filière : Informatique
Option : Système d'information et de decision

Mémoire de Master

Présenté par : Abdelatif Hamdi

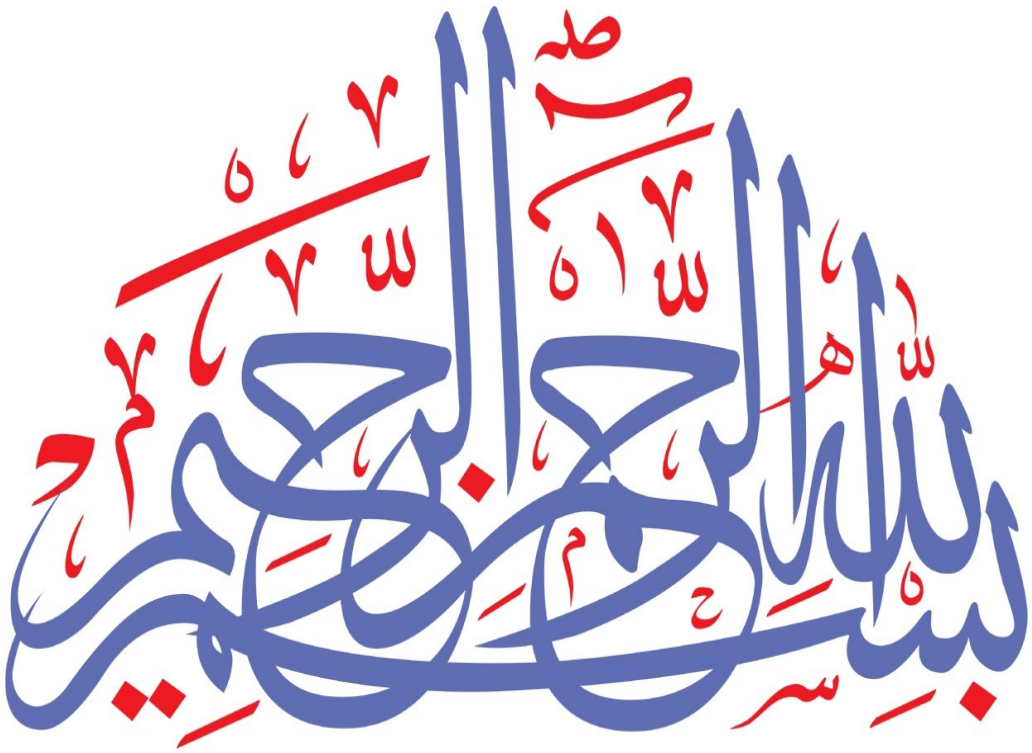
THÈME

Classification des maladies des feuilles de blé à l'aide de techniques d'apprentissage en profondeur

Soutenu publiquement devant le jury composé de :

Mr	Bouzidi Mohamed Ridha	MC(B)	(Université de Laghouat)	Président
Mr	Ouladdjedid Lakhdar Kamal	MC(B)	(Université de Laghouat)	Examinateur
Mr	Bendouma Tahar	MC(A)	(Université de Laghouat)	Examinateur
Mr	Younes Guellouma	MC(A)	(Université de Laghouat)	Encadreur

2023


بسم الله الرحمن الرحيم

Acknowledgments

All my gratitude and thanks to Allah Almighty, who gave me strength, courage and will in light of the current circumstances to develop this work.

I would like to extend my sincere thanks and gratitude to my supervisor, Dr. Younes Guellouma, for his management of this work, and for providing all valuable advice to ensure its success.

I also thank all members of the jury for giving me the honor of evaluating this modest work.

I do not forget my gratitude to all the professors of the Computer Science Department, and thanks to everyone who contributed to the development of this work from near or far.

Abdelatif, June 2023

Dedications

I would like to dedicate this work to my parents, who have always supported and encouraged me to pursue my dreams. Your love and guidance have been invaluable throughout my academic journey. Thanks to them, I am here today, I wish for nothing but to make them proud.

Dedications also go to our dear families, brothers and sisters, who have always been there for me.

Also I dedicate this work to my best friend, who has been my rock through thick and thin. Your unwavering support and friendship have kept me going, even when things got tough.

Thanks everyone.

Abdelatif Hamdi

Abstract

Wheat is one of the most important staple crops in the world, and leaf diseases can significantly reduce its yield and quality. In this study, we propose a deep learning approach for wheat leaf disease classification. We collected a large dataset of wheat leaf images and applied appropriate preprocessing techniques to enhance the quality of the data. Then, we trained a convolutional neural network (CNN) model on the preprocessed dataset and used a genetic algorithm for hyperparameters optimization. Our proposed CNN model achieved an overall accuracy of 98.08% and the other metrics scores ranged from 97% to 100%. We also developed a web app and mobile app for farmers and other stakeholders to easily access and utilize the model. Our work demonstrates the potential of deep learning techniques for accurate and timely diagnosis of wheat leaf diseases, which can ultimately improve crop yield and help sustain food security.

Keywords : **Wheat leaf diseases, deep learning, convolutional neural network, genetic algorithm, hyperparameters optimization, classification, rest API, web app, mobile app, smart farming, food security.**

ملخص

يعد القمح أحد أهم المحاصيل الأساسية في العالم، ويمكن لأمراض الأوراق أن تقلل بشكل كبير من محصولها وجودتها. في هذه الدراسة، نقترح نهجًا تعليميًا عميقًا لتصنيف أمراض أوراق القمح. جمعنا مجموعة كبيرة من صور أوراق القمح وطبقنا تقنيات المعالجة المسبقة المناسبة لتحسين جودة البيانات. بعد ذلك، قمنا بتدريب شبكة عصبية ملتوية على مجموعة البيانات المعالجة مسبقًا واستخدمنا خوارزمية جينية لتحسين المعلمات. حقق نموذج المقترح دقة إجمالية تبلغ 98.08٪ وتراوحت درجات المقاييس الأخرى من 97٪ إلى 100%. قمنا أيضًا بتطوير تطبيق ويب وتطبيق جوال للمزارعين وأصحاب المصلحة الآخرين للوصول بسهولة إلى النموذج واستخدامه. يوضح عملنا إمكانات تقنيات التعلم العميق للتشخيص الدقيق وفي الوقت المناسب لأمراض أوراق القمح، والتي يمكن أن تحسن في نهاية المطاف غلة المحاصيل وتساعد في الحفاظ على الأمن الغذائي.

الكلمات الرئيسية: أمراض أوراق القمح، التعلم العميق، الشبكة العصبية الالتفافية، الخوارزمية الجينية، تحسين المعلمات، التصنيف، واجهة برمجة التطبيقات رست، تطبيق الويب، تطبيق الهاتف المحمول، الزراعة الذكية، الأمن الغذائي.

Résumé

Le blé est l'une des cultures de base les plus importantes au monde, et les maladies des feuilles peuvent réduire considérablement son rendement et sa qualité. Dans cette étude, nous proposons une approche d'apprentissage en profondeur pour la classification des maladies des feuilles de blé. Nous avons collecté un grand ensemble de données d'images de feuilles de blé et appliqué des techniques de prétraitement appropriées pour améliorer la qualité des données. Ensuite, nous avons formé un modèle de réseau neuronal convolutionnel (CNN) sur l'ensemble de données prétraité et utilisé un algorithme génétique pour l'optimisation des hyperparamètres. Notre modèle CNN proposé a atteint une précision globale de 98,08% et les autres scores métriques variaient de 97% à 100%. Nous avons également développé une application Web et une application mobile pour les agriculteurs et les autres parties prenantes afin d'accéder et d'utiliser facilement le modèle. Notre travail démontre le potentiel des techniques d'apprentissage en profondeur pour un diagnostic précis et opportun des maladies des feuilles de blé, qui peuvent finalement améliorer le rendement des cultures et aider à maintenir la sécurité alimentaire.

Mots clés: : **maladies des feuilles de blé, apprentissage en profondeur, réseau neuronal convolutionnel, algorithme génétique, optimisation des hyperparamètres, classification, API de repos, application Web, application mobile, agriculture intelligente, sécurité alimentaire.**

Contents

List of Figures

List of Tables

1 Introduction

Contents

1.1 General Introduction and Problem Statement

Wheat is one of the most important cereal crops in the world, and its production is crucial for ensuring global food security. However, wheat is susceptible to various diseases that can cause significant yield losses and reduce the quality of the harvested grains. One of the most effective ways to manage these diseases is early detection and timely intervention. In recent years, there has been an increasing interest in using machine learning techniques, particularly deep learning, to classify plant diseases and improve crop management.

1.2 Research Objectives

Therefore, we aim to explore deep learning models, specifically focusing on their efficacy in the domain of leaf disease detection. Indeed, we intend to confirm those used in literature in the same context. To ensure optimal performance and accuracy, we will

use some tuning techniques of hyper-parameters. Hyper-parameter optimization plays a crucial role in fine-tuning the model's configuration, enabling us to identify the most effective combination of hyper-parameters that yields the best results. In addition, we use a combination of most used open datasets in literature rather than only considering one.

1.3 Tool Development

With this in mind, we pretend to create a lightweight and user-friendly tool that simplifies the process of leaf disease detection for simple farmers. Indeed, we intend to design a tool that can be easily employed by farmers with minimal technical expertise.

1.4 Thesis Overview

This graduation thesis focuses on the classification of wheat leaf diseases using deep learning techniques. The primary objective of this study is to develop an accurate and reliable model for the early detection and classification of wheat leaf diseases. The study aims to explore the potential of deep learning techniques in addressing the challenges associated with plant disease diagnosis and management.

1.5 Thesis Structure

Besides a general introduction and a conclusion, our master thesis is organized into four chapters. In the first chapter, we will begin with an overview of wheat leaf diseases and their impact on crop production. It will then discuss the current disease detection and classification methods and highlight their limitations. In the second chapter, we will introduce the concept of deep learning and its potential for improving disease classification accuracy. In this chapter, we will also provide the methodology used for the development of the wheat leaf disease classification model. Then the third chapter will discuss the selection of the deep learning architecture, dataset preparation, and model training including the evaluation results and their discussion.

Finally, the last chapter will present a detailed description of the deployment to the production process.

1.6 Conclusion

Overall, this graduation thesis aims to contribute to the development of accurate and reliable methods for wheat leaf disease detection and classification. The findings of this study will be useful for farmers, agronomists, and researchers in improving wheat crop management practices and reducing yield losses due to diseases.

2

Wheat Leaf Diseases

Contents

2.1 Introduction

Wheat (figure 2.1) is one of the most widely cultivated cereal crops in the world, and it is a staple food for a significant proportion of the global population. It is a member of the grass family and is grown in a variety of climates and regions, from temperate to subtropical areas.

Wheat is a valuable source of nutrition, providing carbohydrates, protein, dietary fiber, and various vitamins and minerals. It is also an important crop for livestock feed, as well as for the production of industrial products and biofuels (particularly ethanol).

However, wheat production faces a range of challenges, including disease, climate change, soil degradation, and water scarcity. As a result, there is a growing need for sustainable and innovative approaches to wheat farming and management to ensure the continued availability of this important crop for future generations.

Figure 2.1: Wheat fields

2.2 Wheat Leaf Diseases

Wheat is susceptible to various diseases that can affect its growth, development, and yield. Some common wheat leaf diseases include the following:

2.2.1 Stripe Rust (Yellow Rust)

In the last 15 years, the disease known as wheat stripe rust (figure 2.2) has become the largest biotic limitation to wheat production and threatens the global food supply. Currently, 88% of the world's wheat production is susceptible to wheat stripe rust, leading to global losses of over 5 million tons of wheat with an estimated market value of 1 billion $USD annually [22, 23]. Wheat stripe rust is also known as wheat yellow rust because of its spore color during its asexual infection cycle on wheat. [24]

Figure 2.2: Stripe Rust (Yellow Rust)

2.2.2 Leaf Rust (Brown Rust)

Leaf rust (figure 2.3), caused by Puccinia triticina, is the most common rust disease of wheat. The fungus is an obligate parasite capable of producing infectious urediniospores as long as infected leaf tissue remains alive. Urediniospores can be wind-disseminated and infect host plants hundreds of kilometers from their source plant, which can result in wheat leaf rust epidemics on a continental scale. [25]

Figure 2.3: Leaf Rust (Brown Rust)

2.2.3 Septoria

Septoria tritici blotch (STB) is an important stubble borne foliar disease of wheat (figure 2.4). This disease has increased in importance in the high rainfall cropping regions during the last five years, even though it has been well controlled in Victoria for the last 30 years through the use of partially resistant wheat varieties. The increase in STB in the high rainfall zone has been favored by stubble retention, intensive wheat production, susceptible cultivars, and favorable disease conditions. [26]

Figure 2.4: Septoria

2.3 The current methods for disease detection

There are various methods for disease detection and classification, including traditional methods, newer methods, and artificial intelligence (AI).

2.3.1 Traditional methods

There are many traditional disease detection and classification methods such as visual observation (figure 2.5) and symptom identification, microscopic examination of plant tissues or fungal cultures, and biochemical tests. While these methods are relatively simple and economical, they are often subjective and may require a high level of expertise to evaluate the results correctly. Furthermore, these methods may not always detect low levels of infection, and the accuracy of the results can be affected by environmental conditions, such as temperature and humidity.

Figure 2.5: Visual observation and symptom identification

2.3.2 Newer technologies

The more recent technologies for detecting and classifying diseases include molecular methods like DNA sequencing, polymerase chain reaction (PCR), and serological assays (figure 2.6). These technologies have the potential to be very sensitive and precise, enabling the detection of very low amounts of infection as well as the identification of particular pathogen strains. However, they require specialized equipment and expertise, which can be expensive and time-consuming. In addition, false positive or false negative

results can happen because these techniques might not always be able to differentiate between closely related pathogen species or strains.

Figure 2.6: Polymerase Chain Reaction (PCR)

2.3.3 Machine learning algorithms and artificial intelligence (AI)

Machine learning algorithms and artificial intelligence (AI) have also been used for smart farming and especially in disease detection and classification in recent years (figure 2.7). These methods use large datasets of plant images and disease symptoms to train algorithms to identify specific diseases accurately.

Indeed, by analyzing data on plant diseases such as images from datasets, AI models can accurately classify and identify different diseases. This can be performed throughout many steps. First, a data collection step is needed. High-resolution images or sensor data are collected from plants or crops in the field. Next, an optional operation on the collected data is preprocessing. It is done in order to enhance its quality and extract relevant features. A labeled dataset is then created where each data sample is associated with the corresponding disease class. Experts or plant pathologists annotate the collected data by labeling the images with the correct disease class. Once the model to be defined is trained, it can be used to classify new, unseen data. This classification can help farmers quickly identify diseases in their crops.

However, these methods can be limited by the availability of high-quality training datasets and may not always generalize well to new plant species or environmental conditions.

Figure 2.7: AI in Smart farming

2.4 Conclusion

In this chapter, we learned about some common wheat leaf diseases and the current methods used for disease detection and classification while highlighting the limitations of each method. In the next chapter, we will define the different tools and technologies that will be used in our work.

3

Model Implementation

Contents

3.1 Introduction

Deep Learning is a sub-field of machine learning that works in a manner inspired by the neurons of the brain. The learning method based on an artificial neural network imitates the working of the human brain in processing data and deriving meaningful information to make decisions. [27] Deep learning algorithms learn to recognize patterns in data by training on large datasets and adjusting the network parameters until the output predictions become more accurate.

Deep learning has shown great potential for improving plant disease classification accuracy. By analyzing large datasets of plant images, researchers can train deep learning algorithms to identify patterns and correlations that may not be easily visible to human experts. This can be particularly useful for early detection of plant diseases, which can significantly improve crop yield and reduce the need for harmful pesticides.

One of the main advantages of deep learning is its ability to learn and adapt to new data, which makes it particularly useful for plant disease classification tasks that involve large and complex datasets. Deep learning models can also be trained to make predictions based on multiple sources of data, such as visual symptoms, environmental factors, and genetic data, which can improve the accuracy of disease classification and help farmers make informed decisions about disease management.

Overall, the potential of deep learning for improving plant disease classification accuracy is significant and has the potential to revolutionize the field of agriculture by providing better and more sustainable crop management practices.

3.2 Neural Network (NN)

Neural Networks are the functional unit of Deep Learning and are known to mimic the behavior of the human brain to solve complex data-driven problems.

The input data is processed through different layers of artificial neurons stacked together to produce the desired output.

From speech recognition and person recognition to healthcare and marketing, Neural Networks have been used in a varied set of domains. [1]

3.2.1 Key Components of the Neural Network Architecture

The Neural Network architecture is made of individual units called neurons that mimic the biological behavior of the brain.

Here are the various components of a neuron (figure 3.1).

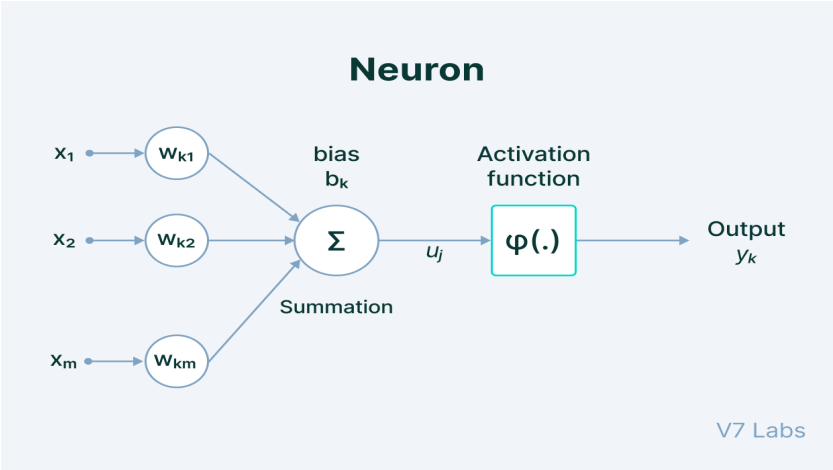


Figure 3.1: Neuron in Artificial Neural Network[1]

3.2.1.1 Input

It is the set of features that are fed into the model for the learning process. For example, the input in object detection can be an array of pixel values pertaining to an image.

3.2.1.2 Weight

The main function of weights is to give importance to those features that contribute more towards learning. It proceeds by introducing scalar multiplication between the input value and the weight matrix. For example, a negative word would impact the decision of the sentiment analysis model more than a pair of neutral words.

3.2.1.3 Transfer function

The goal of the transfer function is to combine multiple inputs into one output value so that the activation function can be applied. It is done by a simple summation of all the inputs to the transfer function.

3.2.1.4 Activation Function

It introduces non-linearity in the working of perceptrons to consider varying linearity with the inputs. Without this, the output would just be a linear combination of input values and would not be able to introduce non-linearity in the network.

3.2.1.5 Bias

The role of bias is to shift the value produced by the activation function. Its role is similar to the role of a constant in a linear function.

3.2.2 Layers

When multiple neurons are stacked together in a row, they constitute a layer, and multiple layers piled next to each other are called a multi-layer neural network.

We've described the main components of this type of structure below (figure 3.2).

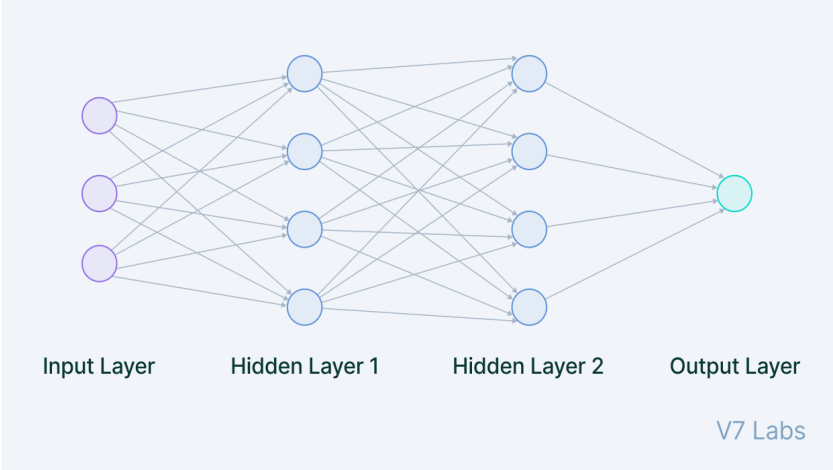


Figure 3.2: Multi-layer neural network[1]

3.2.2.1 Input Layer

The data that we feed to the model is loaded into the input layer from external sources like a CSV file or a web service. It is the only visible layer in the complete Neural Network architecture that passes the complete information from the outside world without any computation.

3.2.2.2 Hidden Layers

Deep learning owes its advancements and capabilities to the hidden layers. These intermediary layers play an important role by performing complex computations and extracting features from the input data.

There can be multiple interconnected hidden layers that account for searching different hidden features in the data. For example, in image processing, the first hidden layers are responsible for higher-level features like edges, shapes, or boundaries. On the other hand, the later hidden layers perform more complicated tasks like identifying complete objects (a car, a building, a person).

3.2.2.3 Output Layer

The output layer takes input from preceding hidden layers and comes to a final prediction based on the model's learning. It is the most important layer where we get the final result.

In the case of classification/regression models, the output layer generally has a single node. However, it is completely problem-specific and dependent on the way the model was built.

In what follows, we consider convolutional neural networks. In fact, CNNs are very suitable to image classification problems, we choose therefore to explore their potential in plant disease classification.

3.3 Convolutional Neural Network (CNN)

A Convolutional Neural Network (ConvNet/CNN) is a Deep Learning algorithm that can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image, and be able to differentiate one from the other. The pre-processing required in a ConvNet is much lower as compared to other classification algorithms. While in primitive methods filters are hand-engineered, with enough training, ConvNets have the ability to learn these filters/characteristics.

The architecture of a ConvNet (figure 3.3)is analogous to that of the connectivity pattern of Neurons in the Human Brain and was inspired by the organization of the

Visual Cortex. Individual neurons respond to stimuli only in a restricted region of the visual field known as the Receptive Field. A collection of such fields overlap to cover the entire visual area. [2]

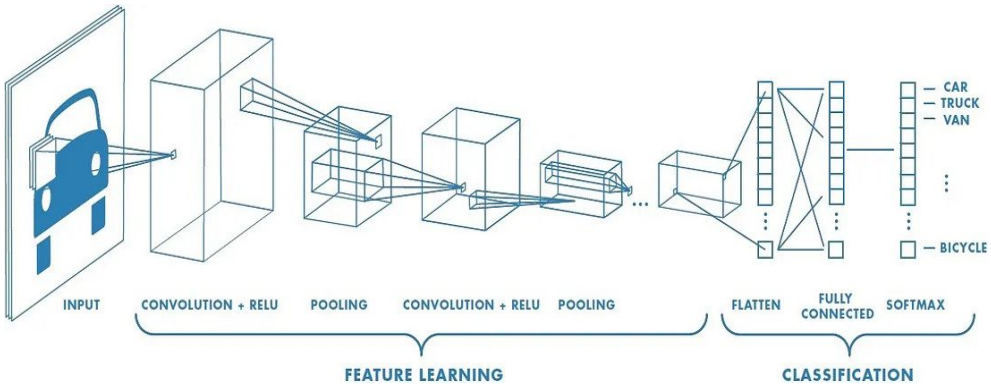


Figure 3.3: Convolutional Neural Network[2]

3.3.1 Types of CNN Layers

CNNs consist of a series of layers that transform the input image data to produce a desired output. The most common types of layers used in CNNs are:

3.3.1.1 Convolutional layers

These layers perform a convolution operation on the input image data using a set of learnable filters (also known as kernels). The filters slide across the image and perform element-wise multiplication with the input pixels to produce a feature map. The resulting feature maps highlight different aspects of the image, such as edges, corners, and textures, that are relevant to the task at hand (figure 3.4). Convolutional layers are typically followed by activation functions such as ReLU to introduce non-linearity in the model.

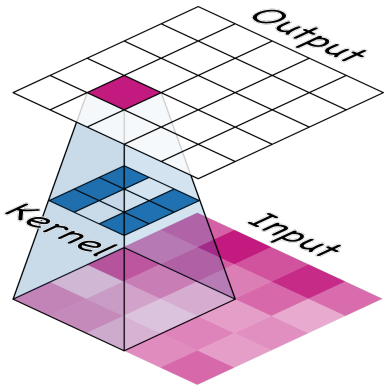


Figure 3.4: convolution operation [3]

3.3.1.2 Pooling layers

These layers downsample the feature maps produced by the convolutional layers by aggregating adjacent pixels. This reduces the spatial size of the feature maps and makes the model more computationally efficient. The most common types of pooling layers are max pooling and average pooling (figure 3.5).

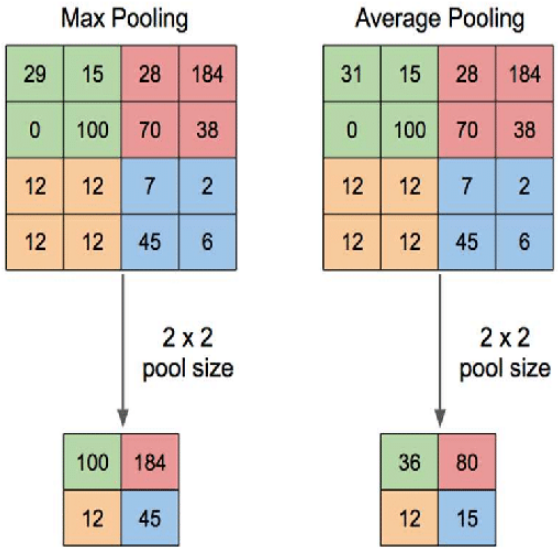


Figure 3.5: Illustration of Max Pooling and Average Pooling [4]

3.3.1.3 Fully connected layers

These layers are similar to the ones used in traditional neural networks. They take the flattened output from the preceding convolutional and pooling layers and produce the final output by applying a set of weights and biases. Fully connected layers are typically used at the end of the network to produce a classification or regression output (figure 4.1).

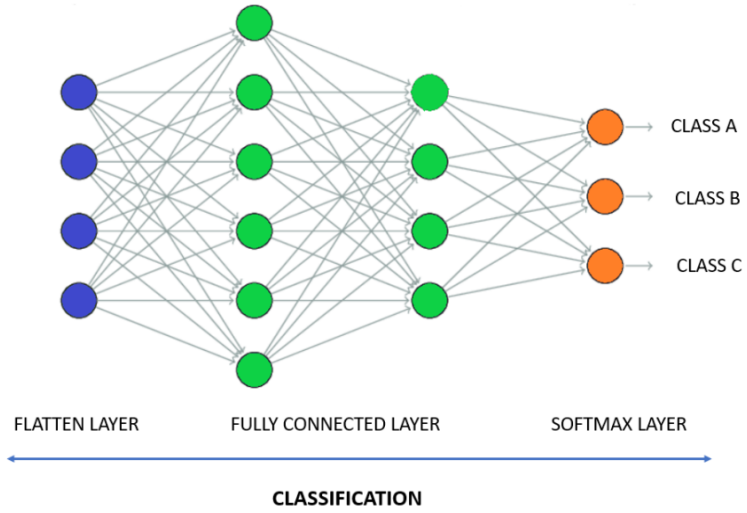


Figure 3.6: The usage of Fully Connected Layers in the classification section of CNNs [5]

3.3.2 How To Choose The Activation Function?

Each task requires the selection of suitable activation function. The softmax function is an activation function applied to the multi-class classification task that normalises output real values from the last fully connected layer to target class probabilities, where each value ranges from 0 to 1 and all values add to 1. The table below summarises typical last layer activation function selections for various types of tasks [21]

Table 3.1: A list of the commonly used activation functions in the last fully connected [21]

Task	Activation Function
Binary classification	Sigmoid
Multiclass single-class classification	Softmax
Multiclass multiclass classification	Sigmoid
Regression to continuous values	Identity

3.3.3 CNN Usage in Plant Diseases Classification (Related Work)

The detection of plant leaf diseases using Convolutional Neural Networks (CNNs) has gained significant attention in the agricultural domain. This section provides a review of recent studies that have explored the application of CNNs in plant disease detection, highlighting the datasets, methodologies, and findings of each study.

3.3.3.1 CNN architectures in Plant Diseases Classification

There are several CNN architectures that have been successfully used in plant disease classification tasks. These architectures include:

1. **AlexNet:** A popular CNN architecture with five convolutional layers and three fully connected layers.
2. **VGGNet:** A deep CNN architecture with up to 19 layers.
3. **ResNet:** A CNN architecture with residual connections that allow for deeper networks.
4. **InceptionNet:** A CNN architecture with multiple parallel paths for extracting features. [28]

3.3.3.2 Studies on CNN Algorithms for Plant Disease Detection

Abade et al. [29] conducted a comprehensive review of CNN algorithms for plant disease detection. They analyzed 121 papers published between 2010 and 2019 and identified PlantVillage as the most widely used dataset. TensorFlow emerged as the most frequently employed framework. Their review encompassed various aspects of

CNN-based approaches, including dataset selection, framework usage, and performance evaluation metrics.

Similarly, Dhaka et al. [30] focused on CNN models for plant disease identification using leaf images. They compared different CNN architectures, preprocessing techniques, and frameworks. The study also discussed the choice of datasets and performance measures for assessing model efficiency. Their findings shed light on the effectiveness of CNN models in addressing plant disease detection challenges.

Nagaraju et al. [31] conducted a review to identify the best datasets, preprocessing approaches, and deep learning (DL) techniques for plant disease diagnosis. They analyzed 84 papers related to DL's applicability in plant disease diagnosis. The review highlighted the limitations of some DL methods in analyzing original images and emphasized the importance of suitable preprocessing techniques for achieving effective model performance.

3.3.3.3 CNN Applications in Agriculture and Weed Monitoring

Kamilaris et al. [32] investigated the broader applications of DL approaches in agriculture and found that DL methods outperformed conventional image processing techniques. The study emphasized the superiority of DL in addressing various agricultural challenges, including plant disease detection.

Fernandez-Quintanilla et al. [33] evaluated weed-monitoring technologies in agricultural fields, encompassing remote sensing and ground-based approaches. The study stressed the significance of weed monitoring for effective weed control. The authors predicted the integration of data acquired from different sensors stored in a public cloud, enabling its utilization in appropriate contexts at optimal times.

3.3.3.4 CNN-Based Approaches for Plant Disease Classification

Lu et al. [34] conducted a review specifically focused on plant disease classification using CNNs. They evaluated the key challenges and proposed solutions in employing CNNs for this task. The study emphasized the need for further research with more complex datasets to achieve more satisfactory results in plant disease classification.

Golhani et al. [35] presented a review paper on hyperspectral data for plant leaf

disease identification. They highlighted existing challenges and potential prospects in utilizing hyperspectral data. The authors also discussed neural network approaches for Spectral Disease Index (SDI) development, emphasizing the importance of testing SDIs on various hyperspectral sensors at the plant leaf scale.

Bangari et al. [36] conducted a review specifically focusing on the detection of potato leaf disease using CNNs. They reviewed several papers and concluded that convolutional neural networks exhibit superior performance in disease detection. The study also highlighted the significant contribution of CNNs in achieving maximum accuracy for disease identification.

3.3.3.5 CNN-Based Tea Leaf Disease Prediction System on Smartphones

Lanjewar and Panchbhai [37] proposed a Convolutional Neural Network (CNN) based tea leaf disease prediction system that operates on smartphones using Platform as a Service (PaaS) cloud infrastructure. The study focused on the development of a portable and accessible solution for tea leaf disease detection. By leveraging the power of CNNs, the authors designed a system that could accurately predict tea leaf diseases by analyzing images captured using smartphones. The proposed system utilized PaaS cloud infrastructure to handle the computational requirements of CNN models. The study demonstrated the potential of CNN-based approaches in enabling real-time disease detection and monitoring in the tea leaf cultivation industry.

3.3.3.6 Compact CNNs with Transfer Learning and Feature Selection for Tomato Leaf Disease Classification

Attallah [38] conducted a study on tomato leaf disease classification using compact Convolutional Neural Networks (CNNs) combined with transfer learning and feature selection techniques. The research aimed to enhance the efficiency and accuracy of disease classification models. By leveraging transfer learning, the study utilized pre-trained CNN models and fine-tuned them for tomato leaf disease classification tasks. Additionally, feature selection techniques were employed to identify the most informative and discriminative features from the input images. The study demonstrated the effectiveness of compact CNN models in achieving high classification accuracy while reducing computational complexity. The findings emphasized the importance of transfer learning and feature selection

for improving the performance of CNN-based disease classification systems.

3.3.3.7 Summary

This section provided a comprehensive overview of the related works in Convolutional Neural Networks (CNNs) for the detection of plant leaf diseases. Various studies were discussed, covering different aspects such as CNN algorithms, preprocessing techniques, datasets, and performance evaluation measures. Additionally, the section highlighted the broader applications of CNNs in agriculture, including weed monitoring, as well as the use of hyperspectral data for plant disease identification. Furthermore, two additional studies were presented, focusing on the development of a CNN-based tea leaf disease prediction system on smartphones and the utilization of compact CNNs with transfer learning and feature selection for tomato leaf disease classification. These studies showcase the application of CNNs in specific plant species and highlight the potential for real-time disease detection, portable solutions, and improved classification accuracy through innovative approaches.

3.4 Conclusion

In conclusion, Chapter 3 provided a thorough overview of neural networks, with a specific focus on convolutional neural networks (CNN) in the context of plant diseases classification. The chapter explored the key components and layers of a neural network architecture, as well as the types of layers found in a CNN. Additionally, it discussed the importance of selecting the right activation function and showcased the practical usage of CNN in plant diseases classification through related work. This chapter laid the groundwork for the subsequent chapters, where we will delve into the implementation of our model.

4

Results and Discussion

Contents

4.1 Introduction

Chapter 4 presents the results and discussions of our implemented plant diseases classification model. It highlights the tools used and outlines the step-by-step process followed for model construction.

In this chapter, we evaluate the performance of our model and analyze the obtained results. We discuss the effectiveness of tools such as Python, TensorFlow, and relevant libraries like NumPy, Matplotlib, and Scikit-learn.

The implementation process encompasses data collection, preprocessing, hyperparameter optimization, CNN model construction, and model evaluation. These steps contribute to the model's accuracy and robustness.

By presenting the results and engaging in discussions, we gain valuable insights into the model's performance, its ability to classify plant diseases accurately, and potential areas for improvement.

This chapter bridges the gap between theory and practice by showcasing the practical implementation of our model. It emphasizes the importance of analyzing and interpreting results to understand the model's performance and practical implications in plant pathology.

In summary, Chapter 4 provides a concise overview of the implemented plant diseases classification model's results and discussions. It delves into the evaluation, analysis, and interpretation of the obtained outcomes, contributing to our understanding of the model's effectiveness.

4.2 Used Tools

In this section we will describe the tools that we used during the development process of our model.

4.2.1 Python

Python is a high-level, interpreted programming language that is known for its simplicity, readability, and ease of use. It was first released in 1991 by Guido van Rossum and has since become one of the most popular programming languages in the world. Python is designed to be easy to learn and has a syntax that emphasizes readability and simplicity,

making it an excellent language for beginners. It is widely used for a variety of purposes, including web development, scientific computing, data analysis, artificial intelligence, and machine learning. Python also has a large and active community of developers who contribute to its development, create new libraries and tools, and provide support to other users.

Figure 4.1: Python Logo [6]

4.2.2 Anaconda

Anaconda is a popular distribution of the Python programming language for scientific computing and data science. It includes a wide range of Python packages, tools, and environments for scientific computing, data analysis, and machine learning. Anaconda provides a powerful platform for data scientists to work with large datasets and perform complex data analysis tasks, and it is widely used in academia, research, and industry. Anaconda also includes a package manager called "conda" that allows users to easily install, update, and manage Python packages and dependencies.

Figure 4.2: Anaconda Logo [7]

4.2.3 Jupyter Notebook

Jupyter Notebook is a web-based interactive computing environment that allows users to create and share documents that contain live code, equations, visualizations, and narrative text. It is widely used in data science and scientific computing for exploring and visualizing data, prototyping machine learning models, and sharing research and analysis results.

Jupyter Notebook supports a wide range of programming languages, including Python, R, Julia, and others, and allows users to write, execute, and debug code within the notebook environment. It also provides a variety of features for interactive data visualization, such as Matplotlib and Bokeh, as well as for data manipulation and analysis, such as Pandas and NumPy.

One of the key features of Jupyter Notebook is that it allows users to document their code and analysis in a single document, which can be easily shared and reproduced by others. Notebooks can be exported to various formats, including HTML, PDF, and Markdown, making it easy to create and share reproducible research and analysis.

Overall, Jupyter Notebook is a powerful and versatile tool for interactive data exploration, analysis, and visualization that has become an essential part of the data science toolkit.

Figure 4.3: Jupyter Logo [8]

4.2.4 Tensorflow

TensorFlow is an open-source machine learning framework developed by Google. It provides a set of tools for building and training machine learning models, including neural networks, deep learning models, and other types of machine learning algorithms.

TensorFlow uses a data flow graph to represent the computational structure of a machine learning model, where nodes represent mathematical operations and edges represent the flow of data between them. This makes it highly scalable and efficient, allowing it to handle large datasets and complex models.

TensorFlow includes a wide range of pre-built tools and modules for data preprocessing, model training, model evaluation, and visualization. It also supports distributed computing, allowing users to run models on multiple machines or in the cloud.

TensorFlow is widely used in industry and academia, with applications in areas such as computer vision, natural language processing, speech recognition, and robotics. It has

also been adopted by many large companies, including Google, Airbnb, and Uber, for building and deploying machine learning models at scale.

Overall, TensorFlow is a powerful and flexible machine learning framework that provides a rich set of tools and capabilities for building and training complex machine learning models.

Figure 4.4: Tensorflow Logo [9]

4.2.5 NumPy

NumPy is a popular open-source Python library for numerical computing. It provides powerful tools for working with arrays, matrices, and other multidimensional data structures, and includes a wide range of mathematical functions for performing operations such as linear algebra, Fourier analysis, and random number generation.

NumPy is built on top of the C programming language, which makes it fast and efficient, even for large datasets. It also includes powerful indexing and slicing capabilities, which allow users to manipulate and extract data from arrays quickly and easily.

NumPy is widely used in data science, scientific computing, and machine learning applications, as it provides a solid foundation for many other Python libraries, including Pandas, Matplotlib, and Scikit-learn. It is also used in other scientific and engineering applications, such as signal processing, image processing, and computational physics.

Overall, NumPy is a powerful and versatile library that is essential for many scientific and data-related tasks in Python.

Figure 4.5: NumPy Logo [10]

4.2.6 Matplotlib

Matplotlib is a popular Python library for creating static, interactive, and animated visualizations in Python. It provides a wide range of tools for creating different types of plots and charts, such as line plots, scatter plots, bar charts, histograms, and heatmaps, among others.

Matplotlib is highly customizable, allowing users to control every aspect of their plots, including the axis limits, labels, colors, line styles, fonts, and more. It also integrates well with other Python libraries such as NumPy, Pandas, and SciPy, making it a popular choice for data visualization and analysis.

In addition to its core functionality, Matplotlib provides several interfaces for creating interactive plots and widgets, including the IPython kernel, Jupyter notebooks, and web applications.

Overall, Matplotlib is a powerful and versatile tool that is widely used in academia, industry, and data science to create high-quality visualizations and communicate complex data insights.

Figure 4.6: Matplotlib Logo [11]

4.2.7 Scikit-learn

Scikit-learn (also known as sklearn) is a popular open-source Python library for machine learning. It is built on top of NumPy, SciPy, and matplotlib, and provides a wide range of tools for data mining and data analysis, including classification, regression, clustering, and dimensionality reduction, among others.

Scikit-learn is designed to be simple and efficient, with a consistent API that makes it easy to use for both beginners and experts. It also includes many useful features such as model selection, data preprocessing, and evaluation metrics, which can help streamline the machine learning workflow.

Overall, scikit-learn is a powerful and versatile tool that is widely used by data scientists and machine learning practitioners in academia and industry.

Figure 4.7: Scikit-learn Logo [12]

4.2.8 Kaggle

Kaggle is the world's largest data science community with powerful tools and resources to help you achieve your data science goals. Kaggle offers a no-setup, customizable, Jupyter Notebooks environment. Access GPUs at no cost to you and a huge repository of community published data and code. Inside Kaggle you'll find all the code and data you need to do your data science work. Use over 50,000 public datasets and 400,000 public notebooks to conquer any analysis in no time. [13]

Figure 4.8: Kaggle Logo [13]

4.3 Model Building Steps

We present the different steps of our model building in this section.

4.3.1 Data collection

The success of any machine learning algorithm depends on the quality and quantity of the data used for training. In this study, we collected two datasets from Kaggle to train our convolutional neural network (CNN) model.
The first dataset, Dataset A, consists of 407 high quality images of wheat leaves divided to the following categories: Stripe rust, Septoria, and Healthy. The images were collected from various sources and were labeled. (Dataset credits [39])

The second dataset, Dataset B, consists of 3679 images of wheat leaves from 3 different categories, Brown rust, Yellow rust, and Healthy. The images were also collected from various sources and were labeled according to their category. The dataset is split into training and validation sets with a ratio of 80:20. (Dataset credits [40])
In addition, we manually checked a random sample of 100 images from each dataset to verify the accuracy of the labels. We found that the labeling accuracy was above 95% for both datasets, indicating that the datasets are suitable for training our CNN model.
Overall, the data collection process was crucial for the success of our study. But collecting high-quality datasets is not enough. we need to apply appropriate preprocessing steps on both datasets to train a robust CNN model that will achieve a high accuracy.

4.3.2 Data Preprocessing

The raw image data collected from Kaggle required preprocessing before it could be used for training the CNN model. The preprocessing steps included resizing the images, normalizing the pixel values, and data augmentation.

Resizing the Images: The raw images in both datasets had different sizes, so we resized all images to a fixed size of 256x256 pixels. This ensures that all images are of the same size, which is required for feeding the data into the CNN model.

Normalizing the Pixel Values: To improve the convergence of the CNN model during training, we normalized the pixel values of the images to be between 0 and 1. This is because the activation functions used in the CNN model work best with normalized inputs.

Data Augmentation: We also applied data augmentation to increase the size of the training dataset and prevent overfitting. we used it in two different ways, the first one is by adding a Data augmentation layer to the architecture of our CNN, and the second one is by using a Python class called ImageDataGenerator. We found that there is no significant difference between the two approaches in terms of performance and accuracy. Still, in terms of coding best practices, it is better to go with the second approach following the microservices logic to make our code clean, organized, and more readable. The Data augmentation techniques used include horizontal and vertical flipping, random rotations, brightness, and zooming.

Splitting the Dataset: We used splitfolders tool to split datasets into training, validation, and test directories using a ratio of 80:10:10 for both datasets and combine

them into a bigger one. This ensures we have enough data to train and test the model's performance.
In addition, we also performed exploratory data analysis (EDA) to understand the distribution of the data and identify any outliers or data imbalances. This allowed us to make informed decisions about the data preprocessing steps and ensure that the CNN model was trained on a well-balanced dataset.
Overall, the data preprocessing steps were crucial for the success of our study. But there is another essestial step. building the architecture of our convolutional neural network.

4.3.3 Hyperparameter Optimization

Hyperparameter optimization plays a crucial role in machine learning, as it significantly impacts the performance of machine learning algorithms. Hyperparameters are parameters that are not learned directly from the data, but rather set by the practitioner. They control the behavior and complexity of the learning algorithm and can have a profound impact on model accuracy. The main objective of hyperparameter optimization is to find the optimal values for these parameters to maximize model performance. So it is an essential part of developing accurate and effective convolutional neural network (CNN) models for wheat leaf diseases classification. In this section, we will talk about the most important hyperparameter optimization techniques, why we choose the genetic algorithm-based technique, and report the results of our experiments.

4.3.3.1 Hyperparameter Optimization Techniques

1. **Grid search**

   Grid search is a sort of "brute force" hyperparameter tuning method. We create a grid of possible discrete hyperparameter values then fit the model with every possible combination. We record the model performance for each set then select the combination that has produced the best performance.

   Grid search is an exhaustive algorithm that can find the best combination of hyperparameters. However, the drawback is that it's slow. Fitting the model with every possible combination usually requires a high computation capacity and significant time, which may not be available. [14]

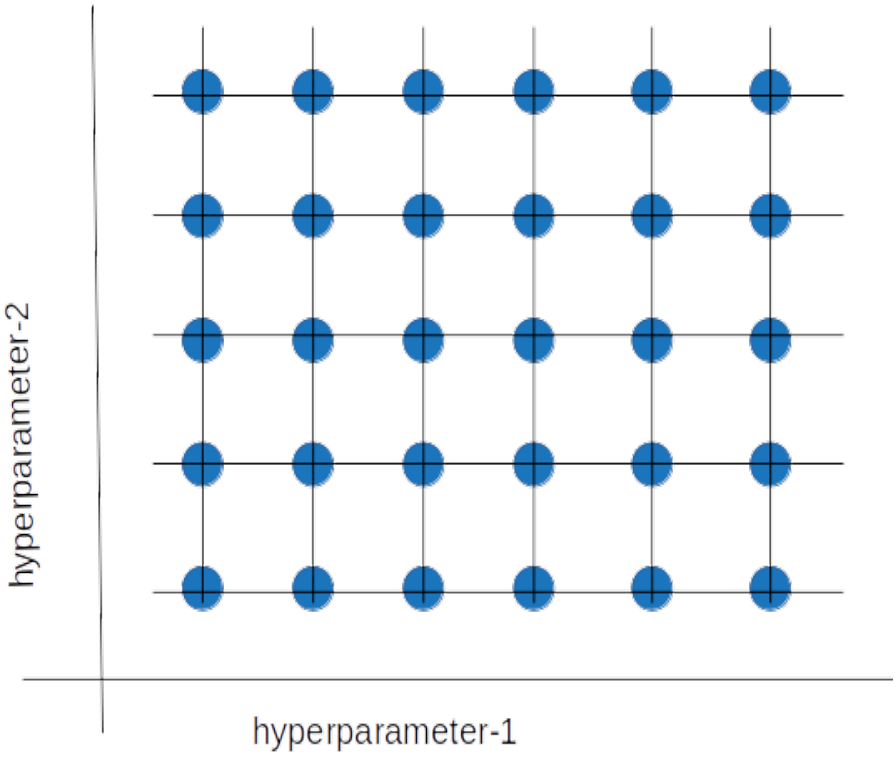


Figure 4.9: Grid Search [14]

2. **Random search**

   The random search method (as its name implies) chooses values randomly rather than using a predefined set of values like the grid search method.

   Random search tries a random combination of hyperparameters in each iteration and records the model performance. After several iterations, it returns the mix that produced the best result.
   Random search is appropriate when we have several hyperparameters with relatively large search domains. We can make discrete ranges (for instance, [5-100] in steps of 5) and still get a reasonably good set of combinations.

   The benefit is that random search typically requires less time than grid search to return a comparable result. It also ensures we don't end up with a model that's biased toward value sets arbitrarily chosen by users. Its drawback is that the result may not be the best possible hyperparameter combination. [14]

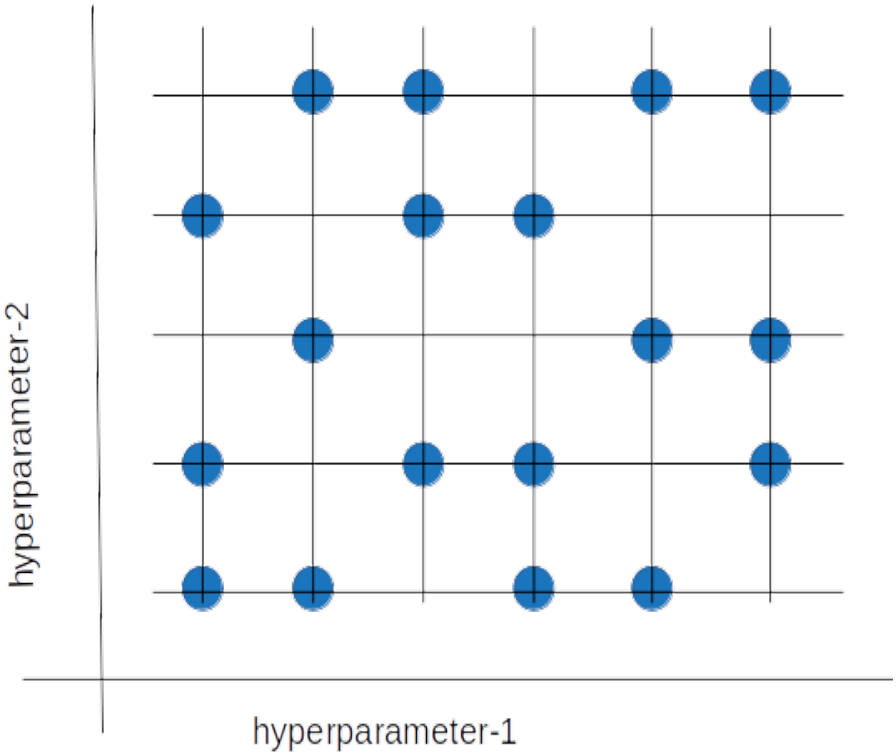


Figure 4.10: Random Search [14]

3. **Bayesian optimization**

   Grid search and random search are relatively inefficient because they often evaluate many unsuitable hyperparameter combinations. They don't take into account the previous iterations' results when choosing the next hyperparameters.

   The Bayesian optimization method takes a different approach. This method treats the search for the optimal hyperparameters as an optimization problem. When choosing the next hyperparameter combination, this method considers the previous evaluation results. It then applies a probabilistic function to select the combination that will probably yield the best results. This method discovers a fairly good hyperparameter combination in relatively few iterations. [14]

4. **Genetic algorithm**

   One of the most prevalent metaheuristic algorithms is the genetic algorithm (GA), which is based on the evolutionary idea that people with the highest survival potential and adaptation to the environment are more likely to survive and pass on their qualities to future generations. The qualities of their parents will be passed on to

the following generation, which may include both good and bad people. Better people will be more likely to live and create more capable children, whereas the worst people will progressively fade away. The individual with the most adaptability will be chosen as the global optimum after multiple generations.

To use GA to hyperparameter optimization issues, each chromosome or person represents a hyper-parameter, and its decimal value reflects the hyper-real parameter's input value in each evaluation. Every chromosome has multiple genes, which are binary digits, and these genes are subsequently subjected to crossover and mutation activities. The population contains all potential values within the initialised chromosome/parameter ranges, whereas the fitness function characterises the parameter assessment metrics.

Since the spontaneous parameter values frequently do not include the optimal parameter values, additional operations, like selection, crossover, and mutation, must be conducted on the well-performing chromosomes to discover the optimums. Chromosome selection is carried out by choosing chromosomes with high fitness function values. To keep the population size constant, chromosomes with high fitness function values are more likely to be passed on to the next generation, where they develop new chromosomes with the best traits of their parents.

Chromosome selection ensures that beneficial traits from one generation are transmitted down to subsequent generations. Crossover is a method of creating new chromosomes by swapping a proportion of genes between chromosomes. Mutation procedures are also employed to create new chromosomes by randomly modifying one or more chromosomal genes. Crossover and mutation processes allow for alternative features in following generations and lessen the likelihood of missing beneficial qualities. [41]

4.3.3.2 The rationale behind choosing the genetic algorithm (GA) approach

The optimization of hyperparameters plays a crucial role in achieving optimal performance in Convolutional Neural Networks (CNNs). However, the high dimensionality of the hyperparameter search space and limited computational resources pose significant challenges. These are the main reasons behind choosing the genetic algorithm (GA) approach as the optimization method for hyperparameter tuning in our case.

1. **Handling High-Dimensional Search Spaces** CNN models typically involve numerous hyperparameters, including learning rates, filter sizes, and network architectures. As the number of hyperparameters increases, the search space expands exponentially, making exhaustive methods like grid search computationally expensive. However, genetic algorithms have the capability to handle high-dimensional search spaces efficiently. By using genetic operators such as mutation and crossover, GAs explore and exploit the search space in a parallel and population-based manner. This ability makes genetic algorithms well-suited for dealing with the challenges of optimizing hyperparameters in CNN models.

2. **Resource Efficiency** Given the limitations of computational resources, particularly when conducting the training process solely on a laptop, the efficient utilization of resources becomes crucial. Genetic algorithms offer the advantage of operating with a smaller population size compared to other optimization techniques like Bayesian optimization. This reduces memory and processing requirements, making it a favorable choice for optimization tasks performed on limited resources.

3. **Parallel Exploration and Exploitation** Genetic algorithms enable parallel exploration of the hyperparameter search space by evaluating multiple candidate solutions simultaneously. This parallelism allows for a more comprehensive search, enabling the algorithm to explore different regions of the search space concurrently. Moreover, by maintaining diversity within the population, genetic algorithms can strike a balance between exploration and exploitation. The algorithm adapts over generations by evolving the population, which promotes exploration of promising regions and convergence towards optimal or near-optimal solutions.

4.3.3.3 Methodology

We used a genetic algorithm to optimize the hyperparameters of our CNN model. A genetic algorithm is a heuristic optimization technique inspired by the process of natural selection. It involves creating a population of candidate solutions and iteratively selecting the best individuals, crossing them over to create offspring, and mutating them to introduce diversity.

We defined the hyperparameters to be optimized, including the activation function, the optimizer, pooling type, and learning rate. The search space for each hyperparameter

was chosen based on long research and experimentation to find the most influential ones. The following table shows the search space for each hyperparameter.

Table 4.1: Search space for hyperparameters

Hyperparameter	Search space
Activation Function	relu, sigmoid, tanh
Optimizer	adam, sgd, rmsprop
Pooling Type	max, average
Learning Rate	0.001, 0.01, 0.1

We used a population size of 5 and a maximum of 4 generations for the genetic algorithm. The fitness function used to evaluate each candidate model was the accuracy on the training dataset. We trained each candidate model for only 5 epochs using the sparse categorical cross-entropy as the loss function and a batch size of 32.

4.3.3.4 Experiments

We trained our CNN model on the same wheat leaf diseases dataset used in previous studies. The dataset consists of more than 5000 images of wheat leaves with three different diseases and a healthy class. We split the dataset into training, validation, and test sets with a ratio of 80:10:10, respectively.

4.3.3.5 Results

The results of our hyperparameter optimization experiments are presented in the following table. We report the accuracy of the last epoch for each generated hyperparameter configuration by the genetic algorithm.

Table 4.2: Results of hyperparameter optimization

Activation Function	Optimizer	Pooling Type	Learning Rate	Accuracy
sigmoid	adam	average	0.001	28.41%
tanh	sgd	average	0.1	74.76%
sigmoid	sgd	average	0.01	28.12%
tanh	adam	max	0.001	88.78%
tanh	rmsprop	max	0.1	85.98%
relu	rmsprop	average	0.1	84.55%
tanh	sgd	average	0.001	74.47%
sigmoid	adam	max	0.001	27.95%
relu	rmsprop	max	0.1	87.25%
sigmoid	adam	average	0.01	28.80%
sigmoid	rmsprop	average	0.001	28.26%
sigmoid	sgd	max	0.1	28.50%
sigmoid	adam	average	0.001	28.24%
relu	rmsprop	average	0.001	83.80%
sigmoid	adam	average	0.01	29.35%
relu	sgd	max	0.01	66.56%
sigmoid	adam	max	0.1	28.77%
relu	**adam**	**max**	**0.001**	**89.31%**
relu	adam	max	0.01	28.89%
relu	adam	max	0.1	27.68%
sigmoid	adam	average	0.01	28.29%
relu	sgd	average	0.001	47.91%

4.3.3.6 Execution Time

The execution time of the genetic algorithm used in this study was approximately two days. The algorithm required an extensive amount of computational resources and iterations to converge on the optimal solution. Due to the complex nature of the problem and the large search space, it was necessary to allow the genetic algorithm sufficient time to explore and evaluate different solutions. The two-day execution time ensured that the

algorithm had ample opportunity to converge and produce reliable results. It is important to note that the execution time may vary depending on the specific hardware and software setup used for running the genetic algorithm.

4.3.3.7 Hardware Configuration

The genetic algorithm used in this study was executed on an HP EliteBook 840 G5 Business Laptop Computer. The laptop was equipped with an 8th Generation Intel Quad-Core i5-8265U processor capable of reaching up to 3.9GHz. It had 16GB of DDR4 RAM and a 512GB PCIe solid-state drive (SSD) for fast and efficient data access. The laptop featured a 14-inch Full HD display with UHD Graphics 620, providing clear and crisp visuals during algorithm execution.

Additionally, the HP EliteBook 840 G5 boasted several convenient features, including a fingerprint reader for enhanced security and a backlit keyboard for comfortable typing in low-light environments. The laptop ran on the Windows 10 Pro operating system, providing a stable and reliable computing environment for executing the genetic algorithm.

The hardware configuration of the HP EliteBook 840 G5 laptop offered sufficient processing power and memory capacity to handle the computational demands of the genetic algorithm. It allowed for efficient execution and enabled the algorithm to explore the solution space effectively.

Figure 4.11: HP EliteBook 840 G5

4.3.3.8 Discussion

The best-performing hyperparameter configuration, had Relu activation function with Adam optimizer, Max pooling type, and a learning rate of 0.001. The accuracy of this configuration was 89% with only 5 epochs.

Compared to the random search method, which requires a large number of hyperparameter configurations to be evaluated, the genetic algorithm-based approach can converge to optimal configurations more efficiently. Our results suggest that a simpler model can achieve high accuracy, and the chosen hyperparameters including the activation function, the optimizer, pooling type, and learning rate are critical for optimal performance. However, the optimal hyperparameter configuration may vary depending on the specific dataset and problem being addressed.

In conclusion, our genetic algorithm-based hyperparameter optimization technique demonstrates its effectiveness in finding an optimal set of hyperparameters for our CNN models used for wheat leaf diseases classification.

4.3.4 Building the CNN Model

After choosing the best-performing hyperparameter configuration, we built a convolutional neural network (CNN) model to classify images of wheat leaf diseases.

Our CNN model consists of six convolutional layers each followed by ReLU activation functions and by a max-pooling layer to reduce the spatial dimensions of the feature maps. The output of the last convolutional layer is flattened and fed into a fully connected layers, which eventually lead to a softmax layer for classification.

The architecture of the CNN model is as follows (figure 4.13):

1. Input Layer: 256x256x3
2. Convolutional Layer 1: 32 filters of size 3x3, activation='relu'
3. Max Pooling Layer 1: pool size=2x2,
4. Convolutional Layer 2: 64 filters of size 3x3, activation='relu'
5. Max Pooling Layer 2: pool size=2x2,
6. Convolutional Layer 3: 64 filters of size 3x3, activation='relu'
7. Max Pooling Layer 3: pool size=2x2,

8. Convolutional Layer 4: 64 filters of size 3x3, activation='relu'
9. Max Pooling Layer 4: pool size=2x2,
10. Convolutional Layer 5: 64 filters of size 3x3, activation='relu'
11. Max Pooling Layer 5: pool size=2x2,
12. Convolutional Layer 6: 64 filters of size 3x3, activation='relu'
13. Flatten Layer
14. Fully Connected Layer: 64 units, activation='relu'
15. Output Layer: softmax (4 units)

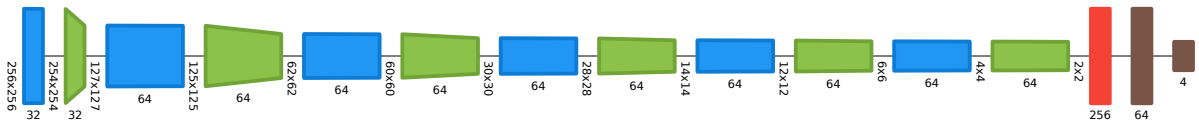


Figure 4.12: Our CNN Model Architecture (Graph)

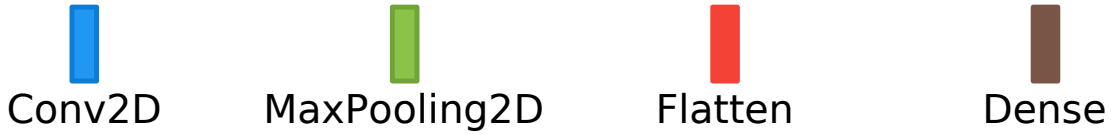


Figure 4.13: Our CNN Model Architecture (Legend)

We used sparse categorical cross-entropy as the loss function and Adam optimizer with a learning rate of 0.001. We trained the CNN model for 50 epochs using mini-batch gradient descent with a batch size of 32.

To evaluate the performance of the CNN model, we used accuracy as the evaluation metric. The CNN model achieved an accuracy of 97.06% on training data, and 98.08% on testing data, indicating that the model is capable of accurately classifying images of wheat leaves.

Overall, building the CNN model required careful consideration of the architecture, loss function, and optimizer to ensure optimal performance. By training the CNN model on the preprocessed datasets, we were able to achieve high accuracy and demonstrate the effectiveness of the CNN model for image classification tasks.

4.3.5 Model Evaluation

In this section, we evaluate the performance of our CNN model for classifying wheat leaf diseases. The evaluation is based on the prediction of the test dataset and using the scikit-learn classification report to assess the model's performance on different metrics such as precision, recall, and F1-score.

Understanding the Classification report through sklearn

A Classification report is used to measure the quality of predictions from a classification algorithm. How many predictions are True and how many are False. More specifically, True Positives, False Positives, True negatives, and False Negatives are used to predict the metrics of a classification report as shown below (figure 4.14). [42]

	precision	recall	f1-score	support
Brown_rust	0.97	0.99	0.98	114
Healthy	0.99	1.00	1.00	151
Septoria	1.00	1.00	1.00	120
Yellow_rust	0.99	0.96	0.98	137
accuracy			0.99	522
macro avg	0.99	0.99	0.99	522
weighted avg	0.99	0.99	0.99	522

Figure 4.14: Classification report of our model using sklearn

The report shows the main classification metrics precision, recall, and f1-score on a per-class basis. The metrics are calculated by using true and false positives, true and false negatives. Positive and negative in this case are generic names for the predicted classes. There are four ways to check if the predictions are right or wrong:

True Negative (TN): when a case was negative and predicted negative

True Positive (TP): when a case was positive and predicted positive

False Negative (FN): when a case was positive but predicted negative

False Positive (FP): when a case was negative but predicted positive

1. **Precision – What percent of your predictions were correct?** Precision is the ability of a classifier not to label an instance positive that is actually negative.

For each class, it is defined as the ratio of true positives to the sum of true and false positives. In other words, it's the accuracy of positive predictions.

$$Precision = \frac{TP}{TP + FP}$$

2. **Recall – What percent of the positive cases did you catch?** Recall is the ability of a classifier to find all positive instances. For each class, it is defined as the ratio of true positives to the sum of true positives and false negatives. In other words, it's the fraction of positives that were correctly identified.

$$Recall = \frac{TP}{TP + FN}$$

3. **F1 score – What percent of positive predictions were correct?** The F1 score is a weighted harmonic mean of precision and recall such that the best score is 1.0 and the worst is 0.0. Generally speaking, F1 scores are lower than accuracy measures as they embed precision and recall into their computation. As a rule of thumb, the weighted average of F1 should be used to compare classifier models, not global accuracy.

$$F1Score = \frac{2 \times (Recall \times Precision)}{Recall + Precision}$$

Results The previous table 4.14 shows the classification report results for our model using a test dataset of 522 images. We achieved an overall accuracy of 98.08% with precision, f1 score, and recall scores ranging from 97% to 100% for each class.

Discussion

Our model achieved an overall accuracy of 98.08%, which indicates good performance in classifying wheat leaf diseases. We can see from the classification report results that our model has good precision, f1 score, and recall scores for each class, ranging from 97% to 100%.

The highest performing class was Septoria, with a precision score of 1.00 and a recall score of 1.00, indicating that our model is very accurate in identifying septoria in wheat leaves. The lowest-performing class was Brown Rust, with a precision score of 0.97 and a recall score of 0.99. This suggests that our model may have a little difficulty with identifying Brown Rust in wheat leaves.

Comparing our results to existing research, we found that our model performed better

than some existing models that achieved an accuracy of around 80% to 97%.

Table 4.3: Related works on wheat leaf disease detection

Reference	Title	Accuracy
[43]	Leaf and spike wheat disease detection & classification using an improved deep convolutional architecture	97.88%
[44]	A High Performance Wheat Disease Detection Based on Position Information	96.4%
[45]	Lightweight Multiscale CNN Model for Wheat Disease Detection	82%
[46]	Image-Based Wheat Fungi Diseases Identification by Deep Learning	94.2%

Limitations

There are two limitations of our evaluation. The first one is that it was based on a limited dataset of only 5198 images. In fact, we believe that the obtained may not represent reality, this is because of dataset representativity bias. Firstly, the process of collecting a dataset may introduce bias if the samples are not randomly selected or if certain subsets of the population are overrepresented or underrepresented. In addition, methods and tools used to collect data may also have biases. Finally, data distribution and human labeling can influence final judgment.

The other one is that we use a very limited hyper-parameter variation for the genetic algorithm during the hyper-parameter optimization phase (Only 4 with a maximum of 3 values for each) and that's because we used the available laptop only for training. In future research, we could evaluate our model on a larger dataset and with a bigger number of hyper-parameters and a wider range of values using a powerful machine to validate our model's performance further.

Thus, we have developed an end-user application that helps the farmer to use our model on one hand and to collect more accurate images directly from local farms to boost the metrics to more real and less biased results on the other hand.

4.4 Conclusion

In this chapter, we talked about the results and discussion part of the study and we found that our CNN model achieved good performance for classifying wheat leaf diseases,

with an overall accuracy of 98.08% and precision, f1 score, and recall scores ranging from 97% to 100%. These results demonstrate the validity of our approach and its potential for accurately identifying different types of wheat leaf diseases.

Note: The source code and the trained models used in this study are available on GitHub at https://github.com/abdelatifhamdi702/Weat-leaf-diseases-detection-notebooks

5

Model Deployment to Production

Contents

5.1 Introduction

After training a CNN model to classify wheat leaf diseases, the next step is to deploy the model for use in real-world applications. In this chapter, we will discuss the deployment of the model in a REST API, as well as the development of a web app and a mobile app that allow end-users to classify images of wheat leaves using the deployed model.

5.2 Model Exportation

In this section, we describe how our wheat leaf disease classification model based on CNN using TensorFlow can be exported for use in a production environment. We will discuss the format and requirements for model exportation, as well as any additional steps that may be necessary for integration into a production system.

5.2.1 The h5 format

The h5 format is a binary data format that is commonly used for storing large scientific datasets, including machine learning models. It is supported by several popular machine-learning libraries, including TensorFlow and Keras.

The h5 format is useful for storing machine learning models because it provides a flexible and efficient way to store large arrays of numerical data. This makes it possible to store the weights and biases of a trained neural network model in a single file, which can then be easily loaded and used for prediction on new data.

The h5 format is particularly useful for deep learning models, which can have millions of weights and biases that need to be stored and loaded efficiently. The format also allows for compression, which can reduce the size of the model file and make it faster to load it, especially for devices with less efficient hardware.

When exporting a machine learning model, you can choose to export it in various formats, including the h5 format. The choice of format will depend on the specific requirements of your production environment and the tools and libraries you are using for deployment. Because we are using TensorFlow and Keras, the h5 format was a good option for exporting and importing our CNN model.

5.2.2 Additional Steps

In addition to exporting the model, there may be additional steps required to integrate it into a production system. For example, we may need to package the model with any required dependencies, such as the TensorFlow library. We may also need to optimize the model for deployment, for example by using TensorFlow Lite to convert the model to a format that can be used on mobile devices.

We recommend consulting with the production system's documentation and requirements to ensure that the model is exported and packaged correctly.

5.3 Building a REST API

To make the model available for use by other applications, we have built a REST API using FastAPI, a modern Python web framework that is designed for building APIs quickly and efficiently. FastAPI is based on the OpenAPI standard, which allows for automatic documentation generation and client library generation.

Figure 5.1: FastAPI Logo [15]

The REST API exposes a predict function that takes an input image and returns a prediction of the image's class (healthy or diseased) along with a percentage probability of the prediction. The predict function loads the saved CNN model and preprocesses the input image before feeding it into the model for prediction. After prediction, the function returns the predicted class and the probability of the prediction.

Indeed, Fast api provides a fast, simple, and lightweight way to help farmers intuitively use the model. By seamlessly integrating with existing agricultural systems, FastAPI empowers farmers to leverage the capabilities of our model, thereby enhancing their decision-making processes and overall productivity.

To handle multiple concurrent requests, we have used the ASGI server implementation

of Uvicorn to serve the REST API. Uvicorn [1] is a fast and lightweight server that is well-suited for high-concurrency applications.

5.4 Hosting the REST API

To make the predict function available for use by the web app and the mobile app, we have hosted the REST API using Render, a cloud platform that provides easy and scalable hosting for modern web applications and APIs. Render offers a simple deployment process with automatic SSL certificate generation, automatic scaling, and built-in metrics and monitoring.

Figure 5.2: Render Logo [16]

5.5 Developing a Web App

To provide an easy-to-use interface for end-users to classify wheat leaves, we have developed a web app using React, a popular JavaScript library for building user interfaces.

Figure 5.3: React Logo [17]

The web app allows users to select an image of a wheat leaf and then calls the predict function in the REST API to classify the image. The app then displays the classification results to the user, including the predicted class and the percentage probability of the prediction.

[1] Unircorn is available at this link https://www.uvicorn.org

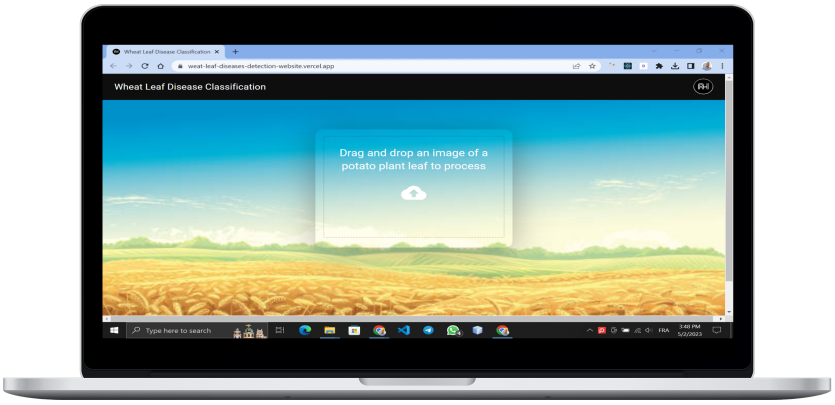


Figure 5.4: The Web App before selecting an image

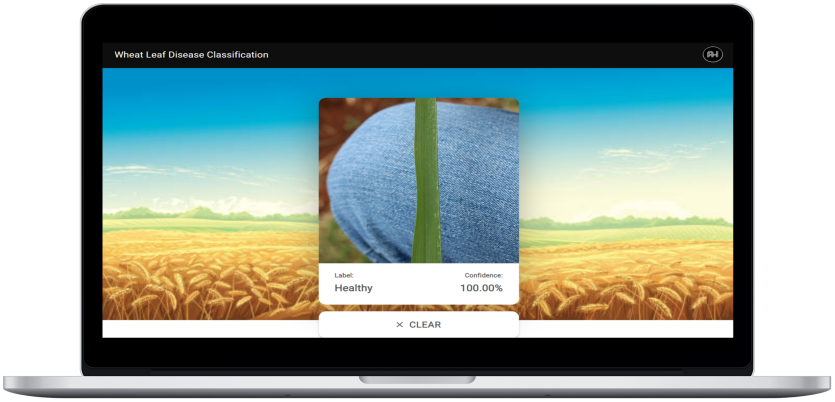


Figure 5.5: The Web App after selecting an image

To interact with the REST API from the web app, we have used the Axios library, a popular HTTP client for the browser and Node.js. Axios provides a simple and elegant interface for making HTTP requests, including GET, POST, PUT, and DELETE requests.

Figure 5.6: Axios Logo [18]

5.6 Hosting the Web App

To make the web app available to end-users, we have hosted it using Vercel, a cloud platform that provides serverless functions and hosting for front-end web applications. Vercel allows us to deploy our React app with zero configuration and no server maintenance. We can also easily scale our app to handle high traffic and improve performance.

Figure 5.7: Vercel Logo [19]

5.7 Developing a Mobile App

To provide an even more convenient way for end-users to classify wheat leaves, we have also developed a mobile app using React Native, a popular framework for building mobile applications for both Android and iOS using React.

Figure 5.8: React Native Logo [20]

The mobile app allows users to either select an image of a wheat leaf from their device's image library or take a picture of a wheat leaf using the device's camera. The app then calls the predict function in the REST API to classify the image and displays the classification results to the user.

To interact with the REST API from the mobile app, we have used the Axios library as well. We have also used the React Native Camera library to access the device's camera and capture images.

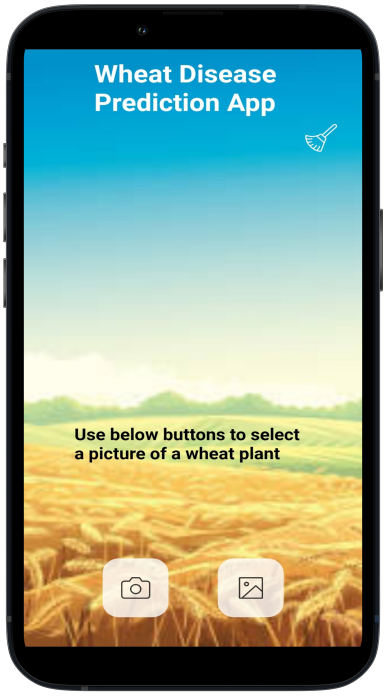


Figure 5.9: The mobile App before selecting an image

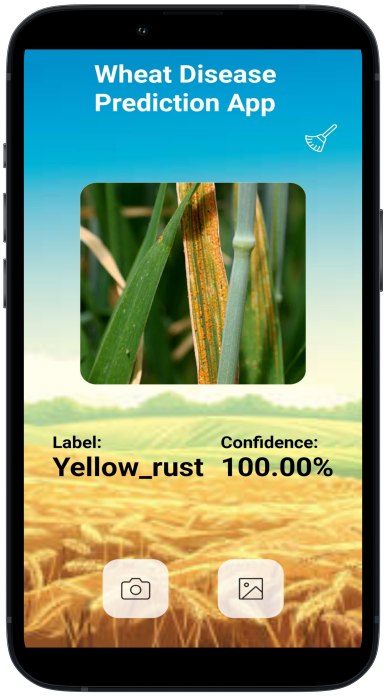


Figure 5.10: The mobile App after selecting an image

5.8 Conclusion

In this chapter, we have discussed the deployment of a CNN model for wheat leaf diseases classification in a REST API, web app, and mobile app. We have exported the model in the h5 format and built a REST API using FastAPI to make the model available for use by other applications. We have developed a web app using React and a mobile app using React Native to provide convenient interfaces for end-users to classify wheat leaves. Finally, we have hosted the web app on Vercel and the REST API on Render to make them available to end-users. With these deployment strategies, our wheat leaf diseases classification model can be easily accessed and utilized by farmers and other stakeholders to detect and diagnose wheat leaf diseases early and accurately, ultimately leading to improved crop yields and food security.

Note: The source code used in this study for the Rest API, Web App and mobile app are available on GitHub at the following links :

Rest API :

https://github.com/abdelatifhamdi702/Wheat-Leaves-Disease-Classification-api

Web App :

https://github.com/abdelatifhamdi702/Weat-leaf-diseases-detection-website

Mobile App :

https://github.com/abdelatifhamdi702/Weat-leaf-diseases-detection-mobile-app

6 Conclusion

Contents

6.1 General Conclusion

In conclusion, our master thesis has explored the use of deep learning techniques for wheat leaf disease classification. Our research has shown that deep learning methods can achieve high classification accuracy and overcome the limitations of current methods for disease detection and classification.

After providing a small overview of the state-of-the-art in leaf detection and recognition using CNNs, we proceeded by constructing our new CNN model, drawing inspiration from the existing literature. Furthermore, we enhanced the overall performance and capabilities of our model through a hyperparameter optimization process, utilizing the available resources at hand. By fine-tuning the hyperparameters, we achieved good improvements in accuracy, resulting in a more accurate and refined model.

By training a CNN model on a dataset of wheat leaf images, we have demonstrated the potential of deep learning to accurately diagnose multiple types of wheat leaf diseases. Furthermore, we have demonstrated the deployment of the model in a REST API, web app, and mobile app, making it easily accessible and convenient for end-users such as farmers and other stakeholders to accurately diagnose wheat leaf diseases in the field,

leading to improved crop yields and food security.

Our work contributes to the growing body of research on deep learning in agriculture and demonstrates its potential for improving crop yields and food security. We hope that our research will inspire further innovation and development in this field and contribute to the advancement of sustainable and efficient food production.

Bibliography

[1] Pragati Baheti. The essential guide to neural network architectures. *Dostupné z: https://www. v7labs. com/blog/neural-network-architectures-guide*, 2021.

[2] Sumit Saha. A comprehensive guide to convolutional neural networks. https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53, 2018. Accessed on 04 14, 2023.

[3] Convolution and relu. https://www.kaggle.com/code/ryanholbrook/convolution-and-relu. Accessed on 04 16, 2023.

[4] Muhamad Yani et al. Application of transfer learning using convolutional neural network method for early detection of terry's nail. In *Journal of Physics: Conference Series*, volume 1201, page 012052. IOP Publishing, 2019.

[5] Fully connected layers in convolutional neural networks. https://indiantechwarrior.com/fully-connected-layers-in-convolutional-neural-networks/. Accessed on 04 16, 2023.

[6] Python. https://www.python.org, note = Accessed on 04 16, 2023.

[7] Anaconda. https://www.anaconda.com. Accessed on 04 16, 2023.

[8] jupyter. https://jupyter.org/. Accessed on 04 16, 2023.

[9] Tensorflow. https://www.tensorflow.org. Accessed on 04 16, 2023.

[10] Numpy. https://www.numpy.org. Accessed on 04 16, 2023.

[11] Matplotlib. https://matplotlib.org/. Accessed on 04 16, 2023.

[12] Scikit-learn. https://www.scikit-learn.org. Accessed on 04 16, 2023.

[13] kaggle. https://www.kaggle.com, note = Accessed on 04 16, 2023.

[14] Juan Navas. What is hyperparameter tuning? https://www.anyscale.com/blog/what-is-hyperparameter-tuning, 2022. Accessed on 05 28, 2023.

[15] Fastapi. https://fastapi.tiangolo.com/. Accessed on 05 02, 2023.

[16] Render. https://render.com/. Accessed on 05 02, 2023.

[17] React. https://react.dev/. Accessed on 05 02, 2023.

[18] Axios. https://axios-http.com/. Accessed on 05 02, 2023.

[19] Vercel. https://vercel.com/. Accessed on 05 02, 2023.

[20] React native. https://reactnative.dev/. Accessed on 05 02, 2023.

[21] Rikiya Yamashita, Mizuho Nishio, Richard Kinh Gian Do, and Kaori Togashi. Convolutional neural networks: an overview and application in radiology. *Insights into imaging*, 9:611–629, 2018.

[22] Colin Wellings. Global status of stripe rust: A review of historical and current threats. *Euphytica*, 179:129–141, 05 2011.

[23] William Cuddy, Jason Beddow, Philip Pardey, Yuan Chai, Terrance Hurley, Darren Kriticos, Han-Joachim Braun, Robert Park, and Tania Yonow. Research investment implications of shifts in the global geography of wheat stripe rust. *Nature Plants*, 1, 09 2015.

[24] Benjamin Schwessinger. Fundamental wheat stripe rust research in the 21st century. *New Phytologist*, 213(4):1625–1631, 2017.

[25] Melvin D Bolton, James A Kolmer, and David F Garvin. Wheat leaf rust caused by puccinia triticina. *Molecular plant pathology*, 9(5):563–575, 2008.

[26] Grant Hollaway. Septoria tritici blotch of wheat. 2007.

[27] Deep learning. https://www.v7labs.com/definitions/deep-learning. Accessed on 04 14, 2023.

[28] Muhammad Hammad Saleem, Johan Potgieter, and Khalid Mahmood Arif. Plant disease classification: A comparative evaluation of convolutional neural networks and deep learning optimizers. *Plants*, 9(10):1319, 2020.

[29] André Abade, Paulo Afonso Ferreira, and Flavio de Barros Vidal. Plant diseases recognition on images using convolutional neural networks: A systematic review. *Computers and Electronics in Agriculture*, 185:106125, 2021.

[30] Vijaypal Singh Dhaka, Sangeeta Vaibhav Meena, Geeta Rani, Deepak Sinwar, Muhammad Fazal Ijaz, and Marcin Woźniak. A survey of deep convolutional neural networks applied for prediction of plant leaf diseases. *Sensors*, 21(14):4749, 2021.

[31] Mamillapally Nagaraju and Priyanka Chawla. Systematic review of deep learning techniques in plant disease detection. *International journal of system assurance engineering and management*, 11:547–560, 2020.

[32] Andreas Kamilaris and Francesc X Prenafeta-Boldú. Deep learning in agriculture: A survey. *Computers and electronics in agriculture*, 147:70–90, 2018.

[33] C Fernández-Quintanilla, JM Peña, Dionisio Andújar, J Dorado, A Ribeiro, and F López-Granados. Is the current state of the art of weed monitoring suitable for site-specific weed management in arable crops? *Weed research*, 58(4):259–272, 2018.

[34] Jinzhu Lu, Lijuan Tan, and Huanyu Jiang. Review on convolutional neural network (cnn) applied to plant leaf disease classification. *Agriculture*, 11(8):707, 2021.

[35] Kamlesh Golhani, Siva K Balasundram, Ganesan Vadamalai, and Biswajeet Pradhan. A review of neural networks in plant disease detection using hyperspectral data. *Information Processing in Agriculture*, 5(3):354–371, 2018.

[36] Sindhuja Bangari, P Rachana, Nihit Gupta, Pappu Sah Sudi, and Kamlesh Kumar Baniya. A survey on disease detection of a potato leaf using cnn. In *2022 Second International Conference on Artificial Intelligence and Smart Energy (ICAIS)*, pages 144–149. IEEE, 2022.

[37] Madhusudan G Lanjewar and Kamini G Panchbhai. Convolutional neural network based tea leaf disease prediction system on smart phone using paas cloud. *Neural Computing and Applications*, 35(3):2755–2771, 2023.

[38] Omneya Attallah. Tomato leaf disease classification via compact convolutional neural networks with transfer learning and feature selection. *Horticulturae*, 9(2):149, 2023.

[39] disease affected and healthy wheat leaf. https://www.kaggle.com/datasets/olyadgetch/wheat-leaf-dataset. Accessed on 10 17, 2022.

[40] Labeled dataset containing wheat diseases. https://www.kaggle.com/datasets/sinadunk23/behzad-safari-jalal. Accessed on 02 10, 2023.

[41] Sourabh Mehta. How to use genetic algorithm for hyperparameter tuning of ml models? https://analyticsindiamag.com/how-to-use-genetic-algorithm-for-hyperparameter-tuning-of-ml-models/, 2022. Accessed on 05 28, 2023.

[42] Muthu Krishnan. Understanding the classification report through sklearn, 2018.

[43] Lakshay Goyal, Chandra Mani Sharma, Anupam Singh, and Pradeep Kumar Singh. Leaf and spike wheat disease detection & classification using an improved deep convolutional architecture. *Informatics in Medicine Unlocked*, 25:100642, 2021.

[44] Siyu Cheng, Haolan Cheng, Ruining Yang, Junyu Zhou, Zongrui Li, Binqin Shi, Marshall Lee, and Qin Ma. A high performance wheat disease detection based on position information. *Plants*, 12(5):1191, 2023.

[45] Xin Fang, Tong Zhen, and Zhihui Li. Lightweight multiscale cnn model for wheat disease detection. *Applied Sciences*, 13(9):5801, 2023.

[46] Mikhail A Genaev, Ekaterina S Skolotneva, Elena I Gultyaeva, Elena A Orlova, Nina P Bechtold, and Dmitry A Afonnikov. Image-based wheat fungi diseases identification by deep learning. *Plants*, 10(8):1500, 2021.