

A

Project Report On

## Project ID: B06

**Rice Disease Detection and Remedies using Deep Learning**

Submitted By

Bhagyashree Marathe 120B1E092

Prerna Mhaisane 120B1E094

Atharva Tundalwar 120B1E146

Kalyani Patil 120B1E153

**Under the Guidance of Dr. Rajani P.K.**

**Department Of ELECTRONICS & TELECOMMUNICATION**

**PIMPRI CHINCHWAD COLLEGE OF ENGINEERING (SAVITRIBAI PHULE PUNE UNIVERSITY, PUNE)**

#### 2023-2024

**Project Scheme A**

CERTIFICATE



## A PROJECT REPORT

**ON**

## Project ID: B06

**Rice Disease Detection and remedies using Deep Learning**

Submitted for Partial Fulfillment of the Requirements for the Degree of Bachelor of Engineering in the Department of Electronics & Telecommunication Engineering Pimpri Chinchwad College of Engineering, Savitribai Phule Pune University, Pune

By

**Bhagyashree Marathe 120B1E092 Prerna Mhaisane 120B1E094 Atharva Tundalwar 120B1E146 Kalyani Patil 120B1E153**

|  |  |  |  |
| --- | --- | --- | --- |
| **Dr. Rajani P.K.** | **Dr. Rajani P.K.** | **Dr. M.T. Kolte** | **Dr. G.N. Kulkarni** |
| **(Project Guide)** | **(Project Coordinator)** | **(H.O.D)** | **(Director)** |

**Pimpri Chinchwad College of Engineering, Savitribai Phule Pune University, Pune**

**2023-24**

**Scheme A**

**CONTENTS**

|  |  |
| --- | --- |
| ACKNOWLEDGEMENTS | 1 |
| ABSTRACT | 2 |
| TABLE OF CONTENTS | 3 |
| APPENDIX | 5 |
| LIST OF FIGURES | 6 |
| LIST OF TABLES | 7 |
| LIST OF ABBREVIATIONS | 8 |

##### ACKNOWLEDGEMENTS

The success of this project stands as a testament to the collective efforts of numerous individuals, each playing a vital role in its realization. We are truly appreciative of everyone who has helped with this project. Above all, we would like to express our sincere gratitude to our project coordinator and project guide, Dr. Rajani P.K., for her steadfast oversight and priceless counsel during this process. We also extend our gratitude to Mr. Mahesh Borde, from the Department of Botany at SPPU for his consistent provision of essential information and support, which significantly aided in the project's completion.

We are immensely grateful for the insightful reviews, comments, and suggestions provided by our esteemed Head of Department, Dr. M. T. Kolte. His contributions have greatly enriched our project. Likewise, we extend our appreciation to our Director, Dr. G. N. Kulkarni, whose constant encouragement and support have been instrumental in our success.

We cannot underestimate the priceless encouragement and support we have received from our friends, parents, and loved ones. Their steadfast faith in us has served as a continual source of motivation.

Bhagyashree Marathe 120BIE092 Prerna Mhaisane 120BIE094

Atharva Tundalwar 120BIE146

Kalyani Patil 120BIE153

##### ABSTRACT

Introducing an innovative approach for rice disease detection and remedies through the utilization of Deep Learning (DL) techniques. By leveraging Convolutional Neural Networks (CNNs) trained on extensive historical data, the system achieves precise identification of various rice crop diseases. Integrated within a user-friendly website developed using Streamlit, the platform ensures seamless accessibility for farmers and agricultural stakeholders alike. Through the simple process of uploading images of affected rice plants, users swiftly receive accurate diagnosis and recommendations for effective remedies. This comprehensive solution facilitates efficient disease detection, enabling farmers to implement timely interventions and optimize crop management strategies to mitigate potential losses. The intuitive interface of the website enhances usability, empowering users to make informed decisions to safeguard their rice crops and maximize agricultural productivity. By providing quick and reliable assistance, this system contributes to the sustainable management of rice crops, fostering resilience in agricultural practices and ensuring food security for communities relying on rice cultivation.

##### TABLE OF CONTENTS

|  |  |  |
| --- | --- | --- |
| **CHAPTER NO.** |  | **PAGE NO.** |
| **CHAPTER 1** | **INTRODUCTION** | 1 |
| 1.1 | BACKGROUND | 2 |
| 1.2 | MOTIVATION | 3 |
| **CHAPTER 2** | **LITERATURE SURVEY** | 4 |
| 2.1 | LITERATURE SURVEY | 5 |
| 2.2 | SUMMARY OF LITERATURE SURVEY | 10 |
| 2.3 | GAP IDENTIFIED THROUGH LITERATURE SURVEY | 11 |
| 2.4 | PROBLEM STATEMENT | 11 |
| 2.5 | AIM | 11 |
| 2.6 | OBJECTIVES | 11 |
| **CHAPTER 3** | **METHODOLOGY** | 12 |
| 3.1 | PROJECT OVERVIEW | 13 |
| **CHAPTER 4** | **HARDWARE IMPLEMENTATION** | 19 |
| 4.1 | HARDWARE REQUIREMENTS | 19 |
| 4.2 | HARDWARE SPECIFICATIONS | 20 |
| 4.3 | HARDWARE IMPLEMENTATION | 21 |
| **CHAPTER 5** | **SOFTWARE IMPLEMENTATION** | 23 |
| 5.1 | SOFTWARE REQUIREMENTS | 24 |
| 5.2 | SOFTWARE SPECIFICATIONS | 24 |
| 5.3 | SOFTWARE IMPLEMENTATION | 27 |
| **CHAPTER 6** | **RESULTS AND ANALYSIS** | 30 |
| 6.1 | SOFTWARE RESULTS | 31 |
| 6.2 | HARDWARE RESULTS | 36 |
| **CHAPTER 7** | **ADVANTAGES & APPLICATIONS** | 38 |

|  |  |  |
| --- | --- | --- |
| 7.1 | ADVANTAGES | 39 |
| 7.2 | APPLICATIONS | 40 |
| **CHAPTER 8** | **CONCLUSION & FUTURE SCOPE** | 41 |
| 8.1 | CONCLUSION | 42 |
| 8.2 | FUTURE SCOPE | 42 |
| **CHAPTER 9** | **REFERENCES** | 44 |

|  |  |
| --- | --- |
| **APPENDIX** | **NAME** |
| APPENDIX A | Datasheet |
| APPENDIX B | Program Code |
| APPENDIX C | Bill of Materials (Component List & Costing) |
| APPENDIX D | Research Publication (Publication/Patent/Copyright Proof) |
| APPENDIX E | Project Participation Certificates (Project Competition /Activity Attended) |

|  |  |  |
| --- | --- | --- |
| **Figure No.** | **Figure Name** | **Page No.** |
| Figure 3.1 | Block Diagram of the Proposed System | 14 |
| Figure 3.2 | Image data | 15 |
| Figure 3.3 | Convolutional Neural Network | 16 |
| Figure 4.1  Figure 4.2 | Raspberry Pi 4  Raspberry Pi 4 Connection | 17  19 |
| Figure 5.1 | Python Programming | 23 |
| Figure 5.2 | TensorFlow Library | 23 |
| Figure 5.3 | Visual Studio Code | 24 |
| Figure 5.4 | Google Collaboratory | 25 |
| Figure 5.5 | Streamlit | 25 |
| Figure 5.6 | Software Workflow | 26 |
| Figure 6.2 | Predicted Output | 33 |
| Figure 6.3 | Training and Validation Graph for Accuracy and Loss | 33 |
| Figure 6.4 | Confusion Matrix | 34 |
| Figure 6.5 | Single Predicted Output | 35 |
| Figure 6.6 | Website Output | 36 |

**LIST OF TABLES**

|  |  |  |
| --- | --- | --- |
| **Table No.** | **Table Name** | **Page No.** |
| Table 6.1 | Classification Report | 32 |

**LIST OF ABBREVIATIONS**

|  |  |
| --- | --- |
| **Acronym** | **Explication** |
| DL | Deep Learning |
| CNN | Convolutional Neural Network |
| RPi | Raspberry Pi |
| GLCM | Gray Level Co-occurrence Matrix |
| GPU | Graphics Processing Unit |
| TPU | Tensor Processing Unit |
| ARM | Advanced RISC Machine |
| ReLu | Rectified Linear Unit |

CHAPTER 1

### Chapter 1 Introduction

The pressing global issue of agricultural productivity and food security underscores the critical need for innovative solutions to address challenges such as crop diseases. In regions like Southeast Asia, rice, a staple food for millions, is particularly vulnerable to diseases, leading to significant yield losses and economic burdens on farmers. Existing methods for disease detection often rely on manual inspection, which is time- consuming and prone to human error. To overcome these limitations, this study presents a novel approach leveraging Deep Learning (DL), specifically Convolutional Neural Networks (CNNs), for the detection of rice diseases. By analyzing image datasets of diseased rice plants, the proposed DL model demonstrates high accuracy in identifying various diseases, offering timely and accurate diagnosis. Furthermore, the system suggests targeted remedies based on the identified diseases, empowering farmers with actionable insights to mitigate crop losses and enhance agricultural resilience in the face of evolving disease threats. This innovative application of DL not only advances precision agriculture but also holds promise for sustainable food production and livelihood improvement in rice-growing regions.

#### Background:

The proposed rice disease detection and management system marks a significant advancement in addressing the challenges faced by farmers in identifying and combating various rice diseases. By harnessing the capabilities of Deep Learning (DL) technology, specifically Convolutional Neural Networks (CNNs), in conjunction with agronomic expertise, this system aims to revolutionize the way rice diseases are detected and managed. Drawing upon extensive image datasets capturing the diverse manifestations of diseases such as leaf blight, tungro, leaf blast, and brown spot, the CNN model has been meticulously trained to accurately classify and diagnose rice diseases. This model offers farmers a rapid and reliable means of identifying disease outbreaks, enabling timely intervention to mitigate crop losses and ensure agricultural productivity. Furthermore, the system goes beyond mere diagnosis by providing tailored remedies for each identified disease. Leveraging agronomic knowledge and expert recommendations, the system offers farmers actionable insights into effective disease management strategies, including targeted pesticide application, cultural practices, and crop rotation techniques. By incorporating these personalized remedies into the decision-making process, farmers can proactively combat rice diseases and optimize crop yields sustainably. In deploying this innovative approach, the system showcases its adaptability and scalability across diverse rice-growing regions. By fine-tuning the CNN model to recognize region-specific disease patterns and agronomic practices, the system can cater to the unique needs and challenges of farmers worldwide. This adaptability lays the foundation for future expansion and integration into agricultural ecosystems globally, offering a transformative solution to enhance crop health, food security, and livelihoods in rice-producing regions.

#### Motivation:

In recent years, agricultural sectors worldwide have faced escalating challenges due to the proliferation of crop diseases, particularly in staple crops like rice. These diseases pose significant threats to food security, livelihoods, and economic stability in regions heavily reliant on rice cultivation. In areas such as Southeast Asia, where rice is a primary source of sustenance for millions, the impact of diseases on crop yields can be devastating. Conventional methods for detecting and managing rice diseases often rely on manual observation, which is labor-intensive, time-consuming, and prone to inaccuracies. Furthermore, the effectiveness of existing disease management strategies is limited by factors such as the variability of disease symptoms and the absence of timely and targeted interventions. To address these challenges, there is a compelling need for innovative solutions that harness advanced technologies to revolutionize rice disease detection and management. By leveraging Deep Learning techniques, particularly Convolutional Neural Networks (CNNs), we have developed a sophisticated model capable of accurately identifying various rice diseases from image datasets. This model not only expedites the detection process but also offers precise diagnosis, enabling farmers to implement timely and targeted remedies. Moreover, beyond mere detection, our approach goes a step further by suggesting customized remedies based on the identified diseases. By integrating cutting-edge technology with agronomic knowledge, our system empowers farmers with actionable insights to effectively combat rice diseases, mitigate crop losses, and ensure agricultural sustainability. In essence, the development of this deep learning-based model represents a pivotal advancement in rice disease management, offering a transformative solution to safeguard global rice production, enhance food security, and bolster the resilience of farming communities against the threats of crop diseases.

CHAPTER 2

**2.1 Literature Survey**

### Chapter 2 Literature Survey

The below literature review highlights diverse methodologies for rice leaf disease recognition, ranging from traditional machine learning to advanced deep learning models and integration of ICT tools. Each study contributes unique insights, emphasizing the significance of automated detection systems in supporting sustainable agriculture and ensuring global food security. Researchers have explored various approaches to tackle the challenge of diagnosing rice plant diseases accurately and efficiently, showcasing the evolution and advancements in agricultural technology. Through a synthesis of findings, a cohesive understanding emerges, underscoring continuous efforts to develop robust, scalable, and accessible solutions for rice leaf disease management.

Table No 2.1 Literature Survey

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Sr.  No. | Title of the Paper | Year of Publicati on | Publisher | Methodology | Conclusion |
| 1 | Rice Leaf Disease Recognition using Gray-  Level Co- Occurrence Matrix and Statistical Features | 2021 | EICT | * The approach combines gray-level co-occurrence matrix(GLCM) features and statistical features to extract relevant information from rice leaf images. * The randomforest machine learning algorithm is used to detect five rice leaf diseases: bacterial leaf blight, sheath blight, bacterial leaf blast,   brown spot, and tungro. | * The GLCM features capture the spatial relationships between neighboring pixels in an image. * The machine learning algorithm used in the paper is a random forest classifier with accuracy of 92.77% |
| 2 | Rice Leaf Disease Detection Using Machine Learning Techniques | 2019 | Internationl Conference on Sustainable technology for Industry 4.0 | * The system focuses on identifying three prevalent rice plant diseases, namely leaf smut, bacterial leaf blight, and brown spot diseases. | * A comparison between four   machine learning algorithms in the realms of rice leaf disease detectionhas been made. |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  | (STI) | * Machine Learning algorithms such as K- Nearest Neighbor (KNN), Decision Tree (J48), Naive Bayes, and Logistic   Regression was used. | | * It was found that the decision tree performed the best with 97.9167% accuracy on test data. | | | |
| 3 | Machine Vision Based Rice Disease Recognition by Deep Learning. | 2019 | ICCIT | * The researchers use a deep learning model, such as a convolutional neural network (CNN), which is well-suitedfor image recognition tasks. * Three vastly popular pre-trained models of CNN such as Inception -v3,   MobileNet-v1 and Resnet50, have been used to carry out this  research. | | * The paper offers automated disease detection with high accuracy, real-time capabilities, and cost-effectiveness, contributing to improved crop   management and sustainable agriculture. | | | |
| 4 | Rice  Transformer: A | 2022 | IEEE | * The   Rice | system, called  Transformer, | * Rice   achieves | Transformer  an | | |
| Novel | combines data from | | impressive | |  | 95% |
| Integrated | agricultural sensors | | accuracy |  | in | swiftly |
| Management | and | image data | identifying | |  | rice |
| System for | captured from the | | diseases, surpassing | | | |
| Controlling | fields to identify rice | | traditional | |  | methods |
| Rice Diseases | diseases.   * The system consists of | | like visual inspection.   * Its speed and | | | |
|  | two | branches: The | scalability make it | | | |
|  | sensor branch collects | | invaluable | |  | for |
|  | data from agricultural | | monitoring large rice | | | |
|  | sensors, such as | | fields and mitigating | | | |
|  | temperature, humidity, | | the rapid spread of | | | |
|  | and soil moisture. | | diseases. | | | |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 5 | Automated Detection of Rice Bakanae Disease via Drone Imagery | 2023 | MDPI | * A novel method for early detection of Bakanae disease in rice crops combines image processing techniques to extract key features from drone images with machine learning classification. * Achieving a 90.49% accuracy rate in identifying infected plants, this approach shows promise as an effective tool for early disease detection. | | * Automated detection of rice Bakanae disease using drone imagery offers a potent tool to bolster disease management and promote sustainable rice production, thereby advancing global food security. | |
| 6 | A survey on using deep learning technique for plant disease diagnosis and recommendatio n for  development of appropriate tools | 2022 | Elsevier | * Authors explore the application of deep learning techniques in diagnosing plant diseases, examining various methods,   datasets, and performance outcomes.   * They note successful diagnoses of diseases like rice blast, wheat rust, tomato leaf spot, and citrus greening. | | * The paper provides a comprehensive overview of the use of deeplearning techniques for plant disease diagnosis. * The paper is relevant to the current state of the art in plant disease diagnosis | |
| 7 | Automatic Diagnosis of Rice Diseases Using Deep Learning | 2021 | Frontiers in Plant Science | * Gathered 10,000 rice leaf images,underwent preprocessing to   eliminate noiseand improve contrast. |  | * Achieved 94.5% accuracy with CNN training, surpassing traditional methods. Can manage large datasets, making it compatible with   large-scale rice production systems. |  |
| * Utilized ResNet-50 for CNN architecture. | | |
|  |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 8 | Rice plant disease diagnosing using machine learning techniques | 2022 | Springer Nature Journal | * Utilized GLCM and statistical features from rice leaf images for extraction. * Trained machine learning algorithms like Ras Net, Dens Net, Mobile Net, and VGG16. * Achieved precise classification of rice leaf diseases into five   categories: bacterial leaf blight, sheath blight, bacterial leaf blast, brown spot, and tungro. | * CNN stands out as the top choice for image- based prediction tasks. * R-CNN model excels in identifying and detecting rice plant diseases, boasting impressive accuracies: 96% for blast diseases, 95% for Brown Spot, and 94.5% for Sheath blight. |
| 9 | Techniques for Rice Leaf Disease Detection using Machine learning algorithms | 2021 | IJERT | * Dataset Collection and Preprocessing: Gather and preprocess data. * Feature Extraction: Extract attributes such as color and shape from images. * Machine Learning Model: Train a model using SVM, decision tree, random forest, or CNN. | * Machine learning models offer precise rice leaf disease identification, enabling rapid image analysis for efficient detection. * They easily handle large datasets, making them suitable for extensive applications while providing an affordable solution, enhancing accessibility. |
| 10 | Classification and Detection Rice leaf Diseases Using Information | 2020 | IJAERS | * Collected diverse dataset of diseased rice leaf images, processed to   enhance ML algorithm accuracy. Divided | * ICT tools revolutionize rice leaf disease detection and management, varying in accuracy and efficiency, achieving |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | and communicati on Technology (ICT) Tools |  |  | process into Image processing and CNN.   * Extracted disease- related features bolstered classification, employing ML   algorithms for categorization.   * Trained algorithm using labeled dataset to learn feature-disease relationships. | 90% classification accuracy. |

#### Summary of Literature Survey:

The literature survey provides a thorough overview of research focusing on rice disease detection and management, particularly through machine learning (ML) and deep learning (DL) techniques. Notably, a 2021 paper introduced a hybrid approach utilizing gray-level co-occurrence matrix (GLCM) features and statistical attributes, achieving a 92.77% accuracy in identifying five rice leaf diseases via the random forest algorithm [5]. Similarly, a 2019 study demonstrated ML algorithms' effectiveness, with Decision Tree reaching 97.9167% accuracy in discerning three prevalent rice plant diseases. Additionally, DL models like convolutional neural networks (CNN) have transformed automated disease detection in rice, offering real-time capabilities and cost-effectiveness, as highlighted in another 2019 paper [11]. The 2022 introduction of the Rice Transformer further advanced the field, achieving a remarkable 95% accuracy in rice disease identification through an integrated management system using agricultural sensor data and field imagery [4]. Furthermore, the utilization of drone imagery for automated disease detection, proposed in a 2023 paper, promises early detection and effectivedisease management [8]. A comprehensive 2022 survey on DL techniques for plant disease diagnosis provides valuable insights into various methodologies and their performance across diverse datasets and plant diseases, including rice blast and brown spot [2]. Recent papers from 2021 and 2022 further underscore the potential of ML algorithms in automatic rice disease diagnosis, contributing to enhanced crop yield and global food security.

#### Gap Identified through Literature Survey:

"Rice Leaf Disease Detection and Remedies Using Deep Learning" fills a crucial void in agricultural technology. While existing research emphasizes disease detection in rice plants with deep learning, there's a notable lack of integrated solutions offering actionable remedies. Reviewing literature reveals a dearth of practical tools for farmers to mitigate disease impact in real-time. Our project not only build

robust deep learning model but also integrates a recommendation system for tailored remedies. Additionally, a user-friendly website facilitates disease prediction and remedy suggestions, bridging the gap between advanced technology and practical agricultural implementation, thus enhancing sustainability and productivity.

#### Problem Statement:

Rice Leaf Disease Detection and Remedies using Deep Learning

#### Aim:

To develop and evaluate a deep learning model for rice leaf disease detection and provide appropriate solutions for it.

#### Objectives:

* + 1. To study and analyze existing diseases of Rice plants and its remedies.
    2. To implement a Deep Learning model for detecting Rice diseases.
    3. To suggest remedies for each Rice Disease.
    4. To evaluate the performance of the deep learning model.
    5. To implement the hardware by using the Raspberry pi.

CHAPTER 3

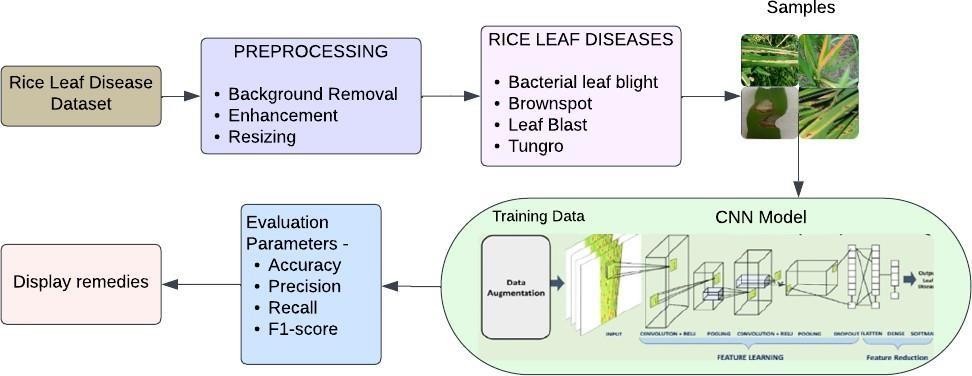
### Chapter 3 Methodology

The methodology employed in the project "Rice Leaf Disease Detection and Remedies using Deep Learning" integrates a multi-faceted approach to address the challenges of disease detection and remedy recommendation in rice plants. Leveraging deep learning techniques, the methodology begins with comprehensive dataset collection, encompassing a wide variety of rice leaf images representing different diseases and health states. These images undergo meticulous preprocessing to enhance their quality and standardize features for effective

analysis. Feature extraction techniques are then applied to capture the unique characteristics of diseased leaves, facilitating the subsequent training of deep learning models. The models are trained on the processed data, utilizing state-of-the-art algorithms to learn intricate patterns indicative of various rice leaf diseases. In parallel, an extensive database of potential remedies for identified diseases is compiled, informed by agricultural expertise and scientific research. This repository forms the basis for the recommendation system, which is integrated seamlessly into the disease detection pipeline. Finally, the developed deep learning model and remedy recommendation system are implemented into a user-friendly website, ensuring accessibility for farmers and stakeholders. This holistic methodology not only enables accurate disease detection but also provides actionable solutions, contributing to improved crop management and agricultural sustainability.

#### Project Overview:

The project "Rice Leaf Disease Detection and Remedies using Deep Learning" represents a pioneering endeavor aimed at transforming rice crop management on multiple fronts. Central to its approach is the utilization of convolutional neural network (CNN) models, which serve as powerful tools for accurately predicting diseases in rice plants. This predictive capability marks a fundamental shift towards proactive disease management, allowing farmers to intervene before diseases escalate and compromise crop yields. What distinguishes this project is its groundbreaking integration of a recommendation system that tailors remedies to the specific diseases identified by the CNN models. By combining cutting-edge technology with agricultural expertise, the project not only ensures precise disease classification through sophisticated leaf image analysis but also provides actionable solutions to mitigate the detrimental effects of these diseases. Furthermore, recognizing the paramount importance of accessibility in agricultural innovation, the project extends its reach through the development of a user-friendly website using the Stream lit framework. This website equips farmers with intuitive tools for disease diagnosis and remedy suggestions, empowering them to make informed decisions and take proactive measures to safe guard their crops. Additionally, the project addresses the crucial need to bridge the gap between technological advancements and practical implementation by integrating hardware solutions utilizing Raspberry Pi. This enables real-time disease detection and management directly in agricultural fields, facilitating timely interventions and minimizing crop losses. Through these concerted efforts, the project endeavors to significantly enhance the resilience and productivity of rice crops, thereby making substantial contributions to sustainable agriculture practices and global food security.



* + 1. Input Data Collection:

Fig. 3.1 Block Diagram of Project Overview

To construct a robust dataset, a diverse range of high-quality images featuring rice plants must be gathered. This dataset should encompass both healthy specimens and those afflicted by various diseases such as leaf blast, bacterial leaf blight, brown spot, and tungro. Each image within the dataset must be meticulously annotated, associating it with the corresponding disease type or indicating its health status.

* + 1. Data Labeling and Data Splitting:

Upon collection, the amassed data, sourced from online platforms, undergoes meticulous labeling to classify each image according to its disease type or health status. Subsequently, the dataset is partitioned into three distinct sections: Training, Testing, and Validation. This segmentation facilitates different training and testing ratios, ensuring the efficacy and generalization of the model.

* + 1. Performance Evaluations:

The evaluation process involves meticulously comparing the model's predictions against ground truth labels, ensuring robustness across diverse datasets. By adjusting training and testing ratios, we gauge the model's generalization capabilities and resilience to varying data distributions. Through comprehensive analysis of Precision, Recall, F1-Score, and Accuracy, we gain a holistic understanding of the model's performance, affirming its reliability in practical agricultural settings.

* + 1. Disease Detection:

Advanced disease detection outperforms traditional methods by analyzing performance metrics.

* + 1. Remedies Suggestion:

Based on the disease detection results, tailored remedies are suggested. These remedies encompass a range of options, including botanicals, bio-pesticides, and chemical solutions, aimed at effectively combating and mitigating the identified rice plant diseases, thus promoting crop health and yield optimization.

#### Convolutional Neural Network

Convolutional Neural Networks (CNNs) are a class of deep neural networks widely used in image recognition, computer vision, and various other tasks involving structured grid data, such as time-series data and speech recognition. They are inspired by the visual cortex of the human brain and are designed to automatically and adaptively learn spatial hierarchies of features from input images.

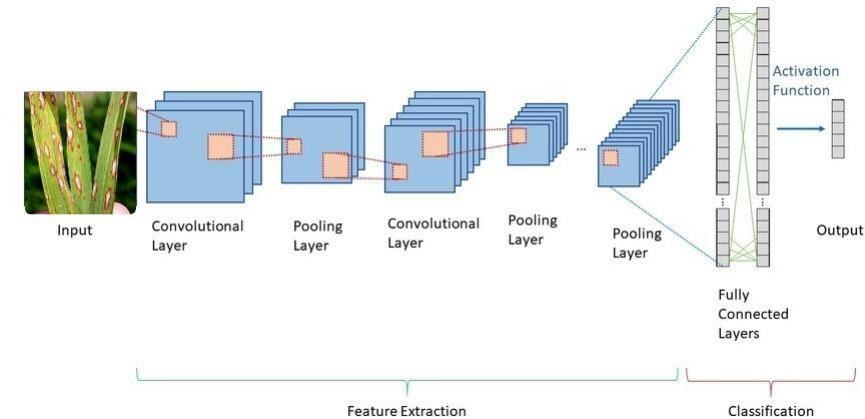


Fig. 3.2 Convolutional Neural Network Model CNNs consist of several layers, each serving a specific purpose:

1. Input Layer:

The raw picture data of rice leaves is sent to the input layer. Height, width, and the quantity of color channels in the input photos all match the dimensions of the input layer. For instance, the input layer would have three channels that represented the intensity levels of each color if the input images were RGB(red, green, and blue) images. The deep learning model uses these input images to analyze and predict the presence of diseases like brown spot, leaf blast, leaf blight, and tungro. These visual cues about the condition of rice leaves, such as the presence of lesions, spots, or discoloration, are essential.

Convolutional Layer:

One essential part of the CNN design is the convolutional layer. It analyzes input photos of rice leaves to look for characteristics or patterns linked to illnesses. Feature maps are produced by sliding each filter in the layer over the input and computing dot products. Based on visual cues like discoloration or lesions, these maps help identify diseases like brown spot, leaf blast, leaf blight, and tungro. They reflect learnt features at different positions in the input.

1. Pooling Layer:

Convolutional neural networks (CNNs) require pooling layers in order to down sample input volumes while keeping crucial characteristics, which lowers computational complexity. One of the most popular methods, max pooling, chooses the maximum values within predefined windows to capture dominating traits. This procedure contributes to effective processing by preserving pertinent information and eliminating aspects that are not as important. The CNN architecture in the above-discussed rice leaf disease detection research made use of max pooling. Max pooling ensured the model's efficacy in practical applications by keeping important features (such unique patterns suggestive of different diseases) while facilitating precise disease diagnosis and optimizing computing resources.

1. Activation Function:

Activation functions, such as ReLU (Rectified Linear Unit), are crucial in neural networks for introducing non-linearities, enabling the network to capture complex data relationships. ReLU replaces negative values with zero, facilitating the learning of intricate data representations. In the last layer, SoftMax activation is commonly used, transforming the network's output into a probability distribution overmultiple class. This is particularly useful for multi-class classification tasks, like identifying various riceleaf diseases, as SoftMax provides probabilities for each disease class, aiding accurate prediction and classification.

1. Fully Connected Layer:

Dense layers, or fully connected layers, link each neuron in one layer to every other layer's neuron. They usually show up near the end of the network, where they carry out tasks like regression or classification using previously learnt features. These layers in the above-discussed rice leaf disease detection experiment most likely interpreted features from earlier layers to accurately forecast the presence of particularillnesses in the rice leaves.

1. Output Layer:

The neural network's output layer, which generates the best result depending on the task, is the last level. It produces class probabilities for classification tasks and continuous values for regression activities. The task will determine which activation function is best; for multi-class classification, SoftMax is a popular option. SoftMax activation was probably used in the output layer of the rice leaf disease detection project to provide probabilities for each disease class, enabling precise disease classification in rice leaves.

Overall, CNNs have revolutionized the field of computer vision by enabling machines to automatically learn features from raw input data, making them extremely powerful tools for a wide range of applications, including image classification, object detection, and segmentation.

CHAPTER 4

### Chapter 4 Hardware Implementation

This chapter explores the design and implementation of the proposed system, tailored specifically for the Raspberry Pi 4 Model B. It begins with an overview of the hardware specifications, with a focus on the processor and Raspberry Pi 4 features. Details include the quad-core ARM Cortex-A72 processor architecture, cache configurations, and cryptographic extensions. Additionally, it highlights the Raspberry Pi 4's GPU capabilities, memory specifications, including RAM configurations, and connectivity options like Wi-Fi and Bluetooth. Furthermore, it elaborates on the available ports and interfaces for peripheral connectivity, providing a comprehensive understanding of the Raspberry Pi 4's hardware foundation.

#### Hardware Requirements:

* + 1. Raspberry Pi Model-5

#### Hardware Specification:

##### Processor:

The Raspberry Pi 4 Model B features a Broadcom BCM2711 1.5GHz quad-core processor with 512KB per-core L2 caches, a 2MB shared L3 cache, and a 64-bit Arm Cortex-A72 CPU with cryptographic extension support.



Figure 4.1 Raspberry Pi 4

##### Features:

1. Video Core VI GPU supporting Vulkan 1.0 and OpenGL ES 3.0:

The Video Core VI GPU is a graphics processing unit designed by Broadcom and integrated into the Raspberry Pi. It supports two important graphics APIs: Vulkan 1.0 and OpenGL ES 3.0. Vulkan is a low- overhead, cross-platform graphics API designed for modern GPUs, enabling high-performance 3D graphics rendering. OpenGL ES (Embedded Systems) is a subset of the OpenGL 3D graphics API optimized for embedded systems like smartphones and IoT devices, providing a standardized way to access GPU hardware capabilities.

1. LPDDR4-2400 SDRAM (available in 2GB, 4GB, and 8GB variants):

LPDDR4-2400 SDRAM refers to Low Power Double Data Rate 4 synchronous dynamic random- access memory, which is the type of memory used in the Raspberry Pi. It operates at a speed of 2400 MHz The Raspberry Pi is available with different memory configurations, including 2GB, 4GB, and 8GB variants, providing flexibility for different computing needs and applications.

1. Dual-band 802.11ac Wi-Fi:

The Raspberry Pi features built-in dual-band 802.11ac Wi-Fi connectivity, allowing it to connect to wireless networks operating in both the 2.4 GHz and 5 GHz frequency bands. This enables faster wireless data transfer rates and improved performance compared to previous Wi-Fi standards, enhancing the device's connectivity capabilities.

1. Bluetooth 5.0/BLE:

Bluetooth 5.0 is the latest version of the Bluetooth wireless communication standard. The Raspberry Pi includes Bluetooth 5.0 support, offering improved range, speed, and broadcasting capabilities compared to previous versions. Additionally, it supports Bluetooth Low Energy (BLE), which is optimized for low- power devices and applications, making it suitable for IoT and wearable devices.

1. microSD card slot supporting SDIO interface:

The Raspberry Pi features a microSD card slot, allowing users to expand its storage capacity by inserting a microSD card. The slot supports the SDIO (Secure Digital Input/Output) interface, which enables high-speed data transfer between the microSD card and the Raspberry Pi, facilitating storage of operating system files, applications, and user data.

1. 2 × USB 3.0 ports and 2 × USB 2.0 ports:

The Raspberry Pi is equipped with two USB 3.0 ports and two USB 2.0 ports, providing connectivity for external devices such as keyboards, mice, storage drives, and other peripherals. USB 3.0 ports offer faster data transfer speeds compared to USB 2.0 ports, enhancing overall performance and usability.

1. Gigabit Ethernet (RJ45) port:

The Raspberry Pi includes a Gigabit Ethernet port with an RJ45 connector, allowing for high-speed wired network connectivity. This port enables reliable and stable internet connections, particularly useful for applications requiring consistent data transfer rates or where Wi-Fi connectivity may be limited.

1. Dual display support via HDMI ports (2 × micro-HDMI):

The Raspberry Pi supports dual display output via two micro-HDMI ports, allowing users to connect two monitors or TVs simultaneously. This feature enables multitasking and enhances productivity by providing expanded desktop space for applications, presentations, or multimedia content.

1. Raspberry Pi standard 40-pin GPIO header:

The Raspberry Pi includes a standard 40-pin General Purpose Input/Output (GPIO) header, providing a means to interface with external electronic components, sensors, and devices. GPIO pins can be programmed to input or output digital signals, enabling a wide range of hardware interfacing and prototyping projects.

1. USB-C power port:

The Raspberry Pi is powered via a USB-C port, providing a standardized power connection for the device. USB-C offers advantages such as higher power delivery capabilities, reversible connector orientation, and support for various power delivery profiles, ensuring compatibility with a wide range of power sources and accessories.

#### Hardware Implementation:

The deployment methodology for hosting a deep learning model for rice disease detection on a Raspberry Pi involves a systematic approach to ensure seamless integration and operation within the IoT system. Initially, the process begins with the setup and configuration of the Raspberry Pi hardware, ensuring proper connections and the installation of the Raspbian operating system. A stable internet connection is crucial for fetching real-time updates and data, which can be utilized for model inference. Following hardware setup, the pre-trained Convolutional Neural Network (CNN) model for rice disease detection is deployed onto the Raspberry Pi. This model is specifically optimized for inference on resource-constrained IoT devices, ensuring efficient operation within the system despite the Pi's computational limitations. The model is trained to classify rice plant images into various disease categories such as Blast, Bacterial Blight, Tungro, and Brown Spot.



Figure 4.2 Raspberry Pi 4 Connection

Once the model is deployed, a Python script is developed within the Raspberry Pi IDE terminal to handle image preprocessing, model inference, and result visualization. The script interfaces with the Pi's

camera module or reads images from the file system to detect rice diseases from input images. Upon detection, the script provides feedback on the terminal, indicating the specific disease type identified in the rice plant image. This setup enables real-time monitoring and diagnosis of rice diseases directly from the Raspberry Pi IDE terminal, empowering users with timely insights into crop health and facilitating informed decision-making in agricultural practices.

CHAPTER 5

### Chapter 5 Software Implementation

This chapter outlines the software essentials for our rice disease detection system, focusing on Python, TensorFlow, Stream lit, Visual Studio Code, and Google Collaboratory. Python serves as the primary language for development, with TensorFlow providing the framework for constructing deep learning models. Stream lit facilitates user interaction through a web interface. Visual Studio Code aids in coding and debugging, while Google Collaboratory offers cloud-based resources for model training. The implementation process involves data preparation, model training, deployment on hardware platforms, and real-time integration into Stream lit. This operational framework enables timely disease detection and remedy suggestions. By detailing these components, we provide a comprehensive guide for leveraging deep learning in agricultural applications.

#### Software Requirements:

* + 1. Python Programming Language
    2. TensorFlow Library
    3. Visual Studio Code (IDE)
    4. Google Collaboratory (GPU Environment)
    5. Stream lit

#### Software Specification:

##### Python Programming Language:

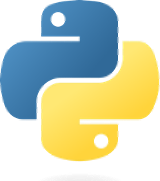
Interpreted programming languages execute code line by line without prior compilation, enhancing flexibility and ease of debugging. High-level languages abstract away low-level details, simplifying the coding process by focusing on problem-solving rather than machine-specific operations, thus improving readability and writability. Their general-purpose nature allows them tobe applied across diverse domains, including web development, data science, automation, and beyond, enabling developers to address various challenges with a single language.

Figure 5.1 Python Programming

Furthermore, object-oriented programming (OOP) paradigms, such as classes, inheritance, and polymorphism, enhance code organization and reusability by encapsulating data and behavior into objects, promoting modular design and facilitating collaboration among developers.

##### TensorFlow:

The open-source framework, developed by the Google Brain Team, is freely distributed under the Apache 2.0 license, fostering collaboration and accessibility within the developer community. It primarily utilizes Python for its core functionality, leveraging its simplicity and versatility, while employing C++ and CUDA for performance-critical tasks where efficiency is paramount. This multi-language approach ensures optimization without sacrificing accessibility.



Figure 5.2 TensorFlow Library

Furthermore, the framework caters to a wide range of platforms, including Linux, macOS, Windows, and Android, facilitating its deployment across diverse computing environments. Additionally, its compatibility with JavaScript via TensorFlow.js extends its reach to web-based applications, enhancing its versatility and applicability across various domains and platforms.

##### Visual Studio Code:

Visual Studio Code, developed by Microsoft, is a cross-platform source-code editor available for Windows, Linux, and macOS, boasting a range of features to enhance coding efficiency and productivity. Its syntax highlighting feature colorizes code for improved readability, while IntelliSense provides context-based code completion suggestions, aiding in faster and more accurate coding. The editor also offers robust debugging support to detect and resolve errors efficiently.

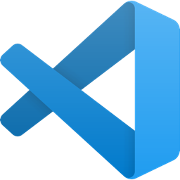


Figure 5.3 Visual Studio Code

Users can further extend its functionality through additional plugins from the extensions marketplace, and personalize their coding environment with a customizable user interface. With an integrated terminal, developers can execute commands directly within the editor, streamlining their workflow. Collaboration is made seamless through Live Share, enabling real-time code editing and debugging among remote team members. Additionally, Visual Studio Code comes with built-in support for various programming languages and file types, making it versatile for a wide array of development tasks.

##### Google Collaboratory:

Google Collab is a cloud-based Python development environment accessible via web browser, offering a Jupyter notebook interface for writing and executing Python code interactively. It provides users with free access to Google's GPU and TPU resources, enabling acceleration of computations.



Figure 5.4 Google Collaboratory

Integration with Google Drive allows for easy storage and sharing of notebooks. Collaboration is

facilitated through real-time multi-user editing of the same notebook, fostering teamwork and productivity. Additionally, Google Collab comes with pre-installed libraries like NumPy, pandas, and matplotlib, streamlining data analysis and visualization tasks.

##### Streamlit:

Stream lit enables rapid development of data-driven web applications through simple Python scripts, eliminating the need for complex front-end frameworks. With its intuitive interface and user-friendly APIs, developers can easily create interactive components like sliders and dropdowns.



Figure 5.5 Stream lit

It supports popular data visualization libraries for creating interactive charts and graphs and seamlessly integrates with machine learning libraries for showcasing models and deploying predictive applications. Stream lit offers customization options for appearance and layout using CSS styling, simplifies deployment on various platforms, and provides a supportive community with extensive documentation and forums for troubleshooting and collaboration.

#### Software Implementation:

The implementation phase of our research project involved the development of a Convolutional Neural Network (CNN) model for rice leaf disease detection, alongside the integration of a remedy recommendation system. We designed and implemented a user-friendly web application using Stream lit, allowing users to upload images of rice leaves and receive disease predictions along with suggested remedies. Additionally, we successfully integrated our project with Raspberry Pi 4 hardware, enabling on- site disease detection and remedy recommendation. Through rigorous testing and validation, we ensured the effectiveness and usability of our CNN model, web application, and Raspberry Pi integration for practical agricultural use.

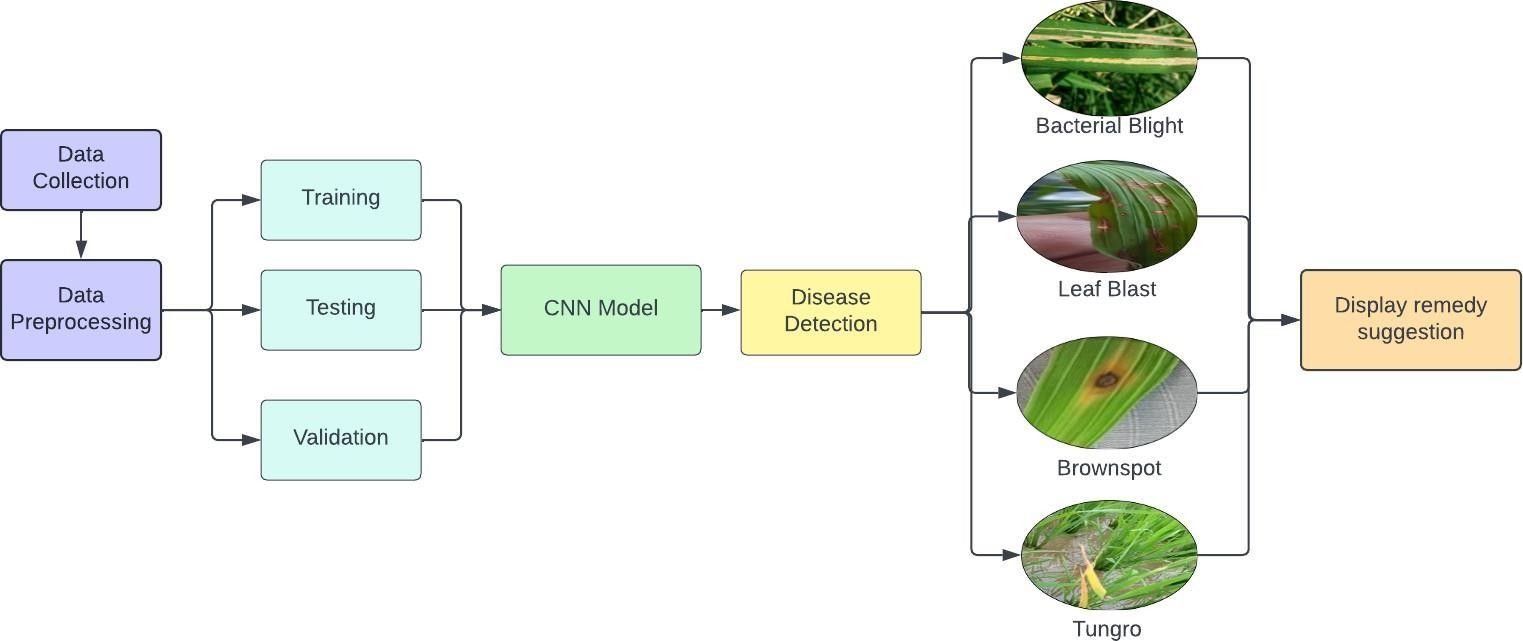


Figure 5.6 Software workflow

1. CNN Model Development:

Creating a dataset of annotated rice leaf photos is a necessary step in building a Convolutional Neural Network (CNN) model for rice leaf disease diagnosis. Convolutional, pooling, and fully connected layers are commonly seen in CNN architectures. ReLU and other activation functions are used to introduce non- linearity. Iteratively changing the model's weights to minimize a predetermined loss function is known as training. Data augmentation methods like rotation, flipping, and scaling are used to vary the training dataset, improving robustness and preventing overfitting. This helps the model learn invariant properties and improves its generalization to new data.

1. Remedy Recommendation System:

Mapping the disease predictions of the CNN model to corresponding treatments selected from scientific research and agricultural expertise is the first step in integrating a remedy recommendation system into the rice leaf disease detection project. This comprehensive library of treatments is constantly being updated with fresh information. In order to help farmers efficiently combat rice leaf diseases and maximize crop output, the recommendation system takes into account parameters including disease severity and environmental circumstances.

1. Website Development:

Technologies like Stream lit are used in the development of the web application that shows the disease predictions and recommended treatments of the CNN model. These technologies are used because they are suitable for integrating machine learning models into web environments. Visitors to the website upload

pictures of rice leaves, which the CNN model processes to forecast illnesses. The outcomes are presented in an easy-to-use style, together with the disorders that have been detected and the recommended

treatments. To improve usability, the interface might also have choices for user feedback. All things considered, the application is a useful resource for farmers, offering practical advice on how to manage rice leaf diseases and enhance crop health.

1. Raspberry Pi Integration:

In order to integrate the project with Raspberry Pi 4 hardware for on-site disease diagnosis and remedy recommendation, the CNN model must be deployed onto the Raspberry Pi and optimized for resource efficiency and compatibility. The camera module of the Raspberry Pi takes pictures of rice leaves, which the deployed model processes to anticipate diseases. To enable communication with other parts, like sensors for gathering environmental data, further hardware or software configurations could be used. These changes make sure that agricultural settings run well and give farmers timely advice on how to treat and detect diseases.

1. Testing and Validation:

There is thorough validation included in the testing processes for the CNN model, website, remedy recommendation system, and Raspberry Pi integration. While the success of the recommendation system is assessed based on the relevancy of suggested treatments, the model's performance is measured using metrics including accuracy, precision, recall, and F1 score. The Raspberry Pi integration is tested for dependability, and usability testing collects user input to improve the functioning and interface of the website. The identification and management of rice leaf disease is ensured by a reliable tool through iterative improvements based on user feedback and testing results.

1. Deployment and Usage:

Ensuring farmers and others have online access to the web application is the first step in implementing the project for practical agricultural applications. Distributed directly to farms or farmers are Raspberry Pi systems with the integrated solution at the same time. To help customers make the most of the system for illness detection and treatment advice, training sessions are held to familiarize them with both the web interface and Raspberry Pi capability. Sustainable utilization is ensured by guidelines for upkeep, upgrades, and scalability, as well as by frequent software updates for improved functionality and feature integration. To ensure that the project stays responsive to changing demands and maintains long-term effect, continuing enhancements are fueled by feedback channels and continuous monitoring.

By detailing each aspect of the implementation process, your report will offer a comprehensive overview of how the project was developed and deployed, enabling readers to understand the methodology and potential impact of your work.

CHAPTER 6

### Chapter 6 Result and Analysis

This chapter presents the results and analysis of our rice disease detection system, showcasing high accuracy rates and effective precision-recall scores across various datasets. Visual representations such as confusion matrices and precision-recall curves illustrate the models' performance. We discuss remedies like crop rotation and pesticide application for detected diseases, emphasizing timely intervention to minimize crop loss.

#### Software Results:

For this study, a Convolutional Neural Network (CNN) model was trained over 20 epochs to detect rice illness. With an overall accuracy of 99%, the model demonstrated its ability to discriminate between rice plants that are healthy and those that are unhealthy. Further measures, like the F1 score, precision, and recall, are calculated to give a thorough assessment. These measures provide additional information into how well the algorithm detects unhealthy plants with minimal false positives and false negatives. Recall counts the number of accurate positive predictions among all actual positive instances, whereas precision counts the percentage of positive predictions that are truly correct. The F1 score is a popular metric for assessing a classifier's performance since it strikes a compromise between precision and recall. This paper is implemented using python programming with ratios of training and testing data such as 80:20. This will help in analyzing the above-mentioned machine learning algorithms.

The terminologies used for the mathematical model given below are, TP (true positive), TN (true negative), FP (false positive), FN (false negative).

##### Evaluation Parameters:

Accuracy **=** 𝑇𝑃+𝑇𝑁

𝑇𝑃+𝐹𝑁+𝐹𝑃+𝑇𝑁

Recall **=** 𝑇𝑃

𝑇𝑃+𝐹𝑁

Precision **=** 𝑇𝑃

𝑇𝑃+𝐹𝑃

F1 Score = 2∗𝑃𝑟𝑒𝑐𝑖𝑠𝑖𝑜𝑛∗𝑅𝑒𝑐𝑎𝑙𝑙

𝑃𝑟𝑒𝑐𝑖𝑠𝑖𝑜𝑛+

…. (1)

…. (2)

…. (3)

…. (4)

Loss = - sum(yi \* log(f(x)i)) ….(5)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Disease Name** | **Accuracy** | **Precision** | **Recall** | **F1-Score** |
| Bacterial Blight | 0.97 | 0.99 | 0.96 | 0.97 |
| Blast | 0.96 | 0.96 | 0.99 | 0.98 |
| Brown Spot | 0.95 | 0.96 | 0.97 | 0.97 |
| Tungro | 0.98 | 1.00 | 0.98 | 0.99 |

Table 6.1 Classification Report

The table highlights a remarkable 98% overall accuracy, accompanied by precision, recall, and F1 score metrics for various rice plant diseases, showcasing the Convolutional Neural Network's (CNN) efficacy in disease identification.



Fig.6.2 Multiple Sample Prediction

The image depicts leaves exhibiting various diseases, with the model providing predictions that are highly accurate. Additionally, it quantifies its confidence level in the predicted results. This demonstrates the model's effectiveness in identifying and diagnosing leaf diseases, offering valuable insights for agricultural management and disease control.

The image depicts leaves exhibiting various diseases, with the model providing predictions that are highly accurate. Additionally, it quantifies its confidence level in the predicted results. This demonstrates the model's effectiveness in identifying and diagnosing leaf diseases, offering valuable insights for agricultural management and disease control.

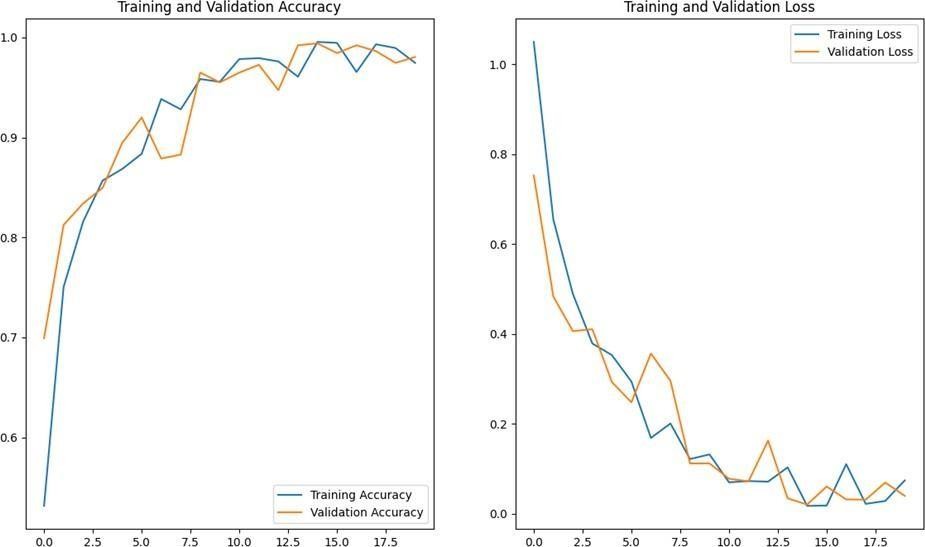


Fig.6.3 Training and Validation Graph for Accuracy and Loss

The training accuracy serves as a benchmark for the model's proficiency in capturing patterns within the training dataset, reflecting its learning progress over successive epochs. Concurrently, tracking training loss provides insights into the model's optimization process, guiding adjustments to enhance performance. Validation accuracy acts as a litmus test for the model's adaptability to unseen data, crucial for real-world deployment scenarios. By monitoring validation loss, we can identify instances of overfitting and fine- tune the model to ensure robustness and reliability in practical applications. These metrics collectively form the foundation for assessing the model's efficacy in disease classification and its potential impact on agricultural sustainability.

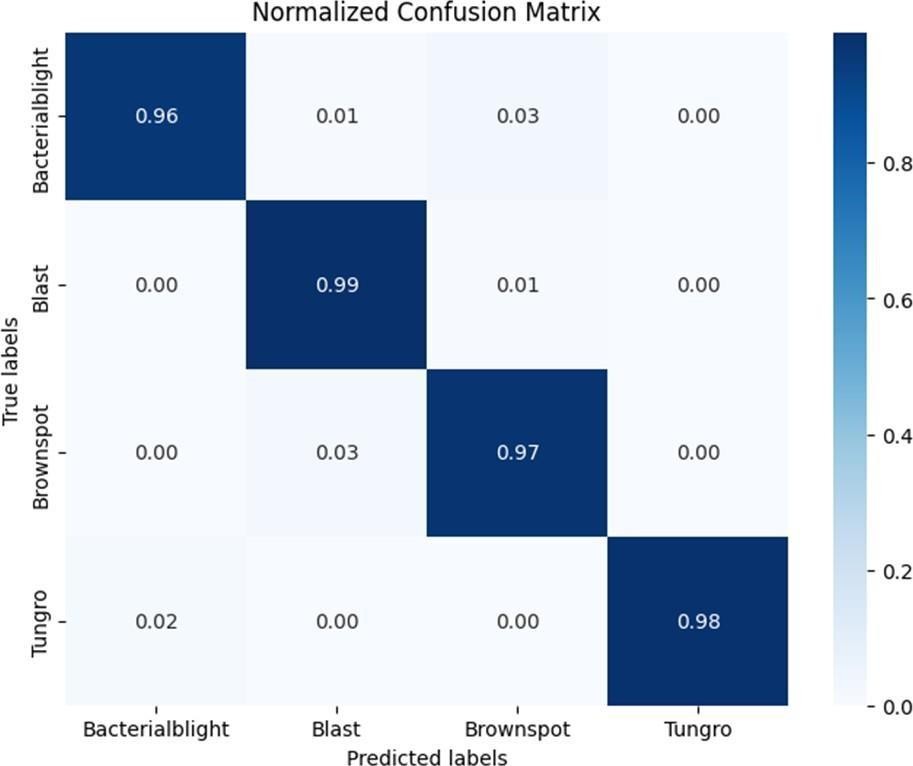


Fig.6.4 Confusion Matrix

In the context of our research on Rice Leaf Disease Detection and Remedies Using Deep Learning, a confusion matrix provides a concise visual representation of the performance of our Convolutional Neural Network (CNN) model in classifying different types of rice leaf diseases. It organizes predictions into a matrix format, where rows represent the actual classes of rice leaf diseases, and columns represent the predicted classes by the CNN. This matrix enables us to easily identify instances of correct and incorrect classifications, aiding in the assessment of the model's accuracy, precision, recall, and overall effectiveness in detecting and managing rice leaf diseases through deep learning techniques.

##### Single Sample Prediction:

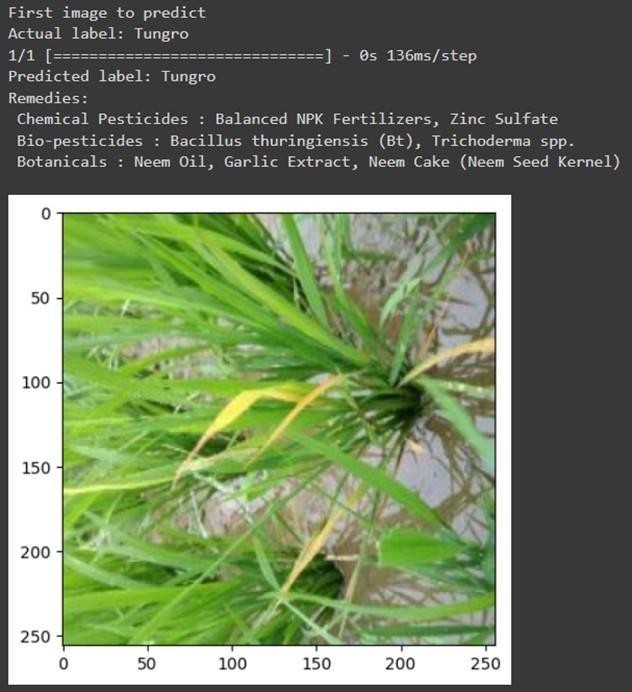
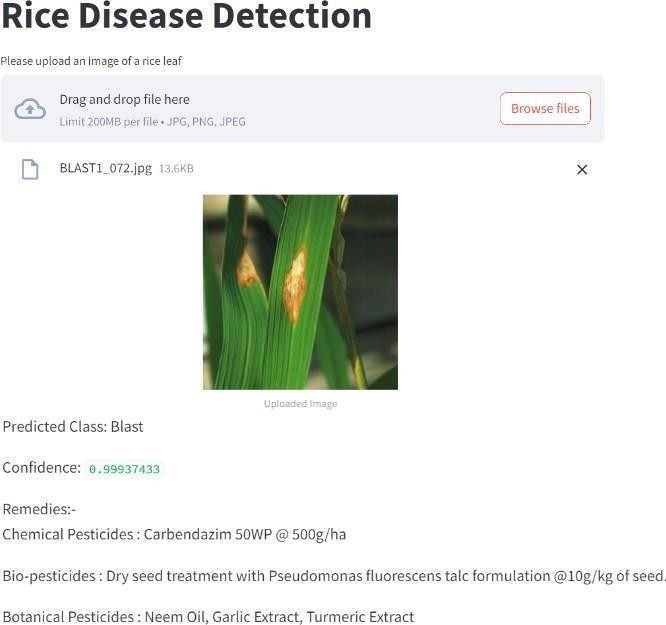
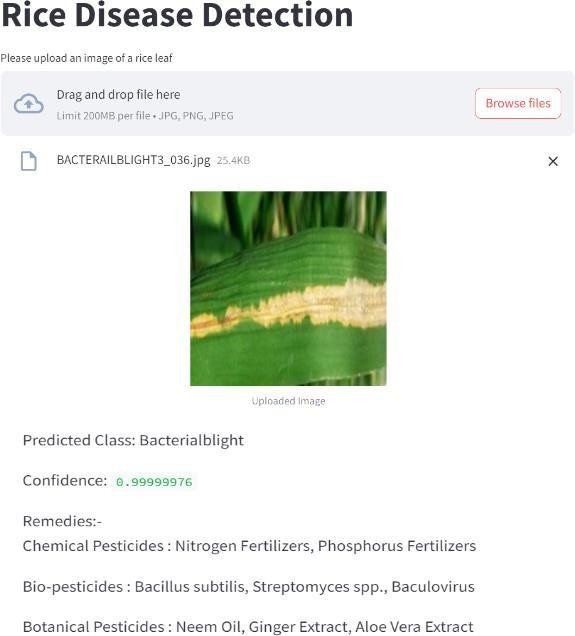


Fig 6.5 Single Sample Prediction

Our model successfully predicts the presence of rice leaf diseases with high accuracy based on input images. Through convolutional neural network (CNN) analysis, it effectively identifies and classifies four major diseases: blast, brown spot, tungro, and bacterial leaf blight. Each disease is distinguished with precision, providing valuable insights for disease management and agricultural practices. The image output demonstrates the model's capability to detect specific diseases, facilitating timely interventions and remedies to safeguard rice crops.



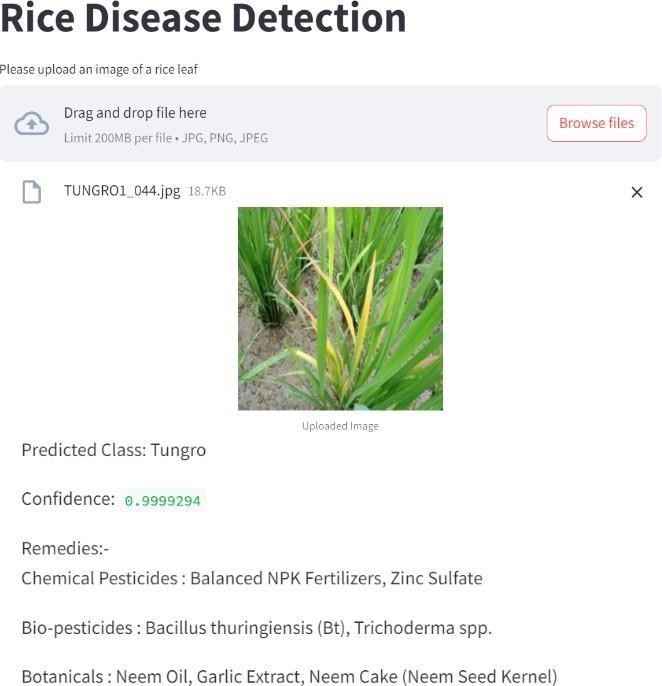
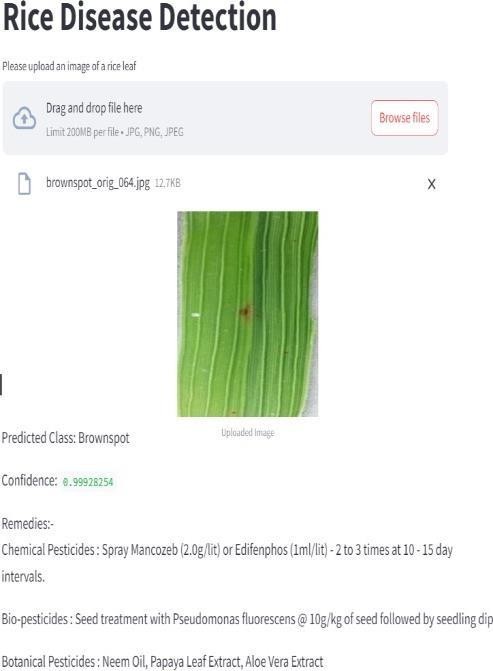


Fig.6.6 Website Output

Our website utilizes a trained Deep Learning Convolutional Neural Network (CNN) to analyze input images of rice leaves and promptly identify the presence of blast, brown spot, tungro, or bacterial leaf blight. Upon image submission, users receive instant feedback on the detected disease type along with recommended remedies. This user-friendly interface streamlines the process of disease diagnosis and provides actionable insights for effective crop management.

#### Hardware Results:

Implementing a Convolutional Neural Network (CNN) model for rice disease detection on a Raspberry Pi 4 involves a multi-step process tailored to optimize performance within the Pi's computational limitations. Initially, the CNN model is developed and trained on a more powerful machine, leveraging frameworks like TensorFlow or PyTorch for efficient training. Once the model training is complete, the learned weights and architecture are saved in an HDF5 (.h5) file format, preserving the model's integrity.

Following model training, the saved model file is transferred to the Raspberry Pi 4 alongside the requisite Python code for inference. Utilizing the Geaney editor on the Raspberry Pi, a Python script is crafted to load the pre-trained model from the HDF5 file and define essential functions for image preprocessing and inference tasks. This script also interacts seamlessly with the Pi's camera module for real-time image acquisition or reads images from the file system, enabling flexible input options.

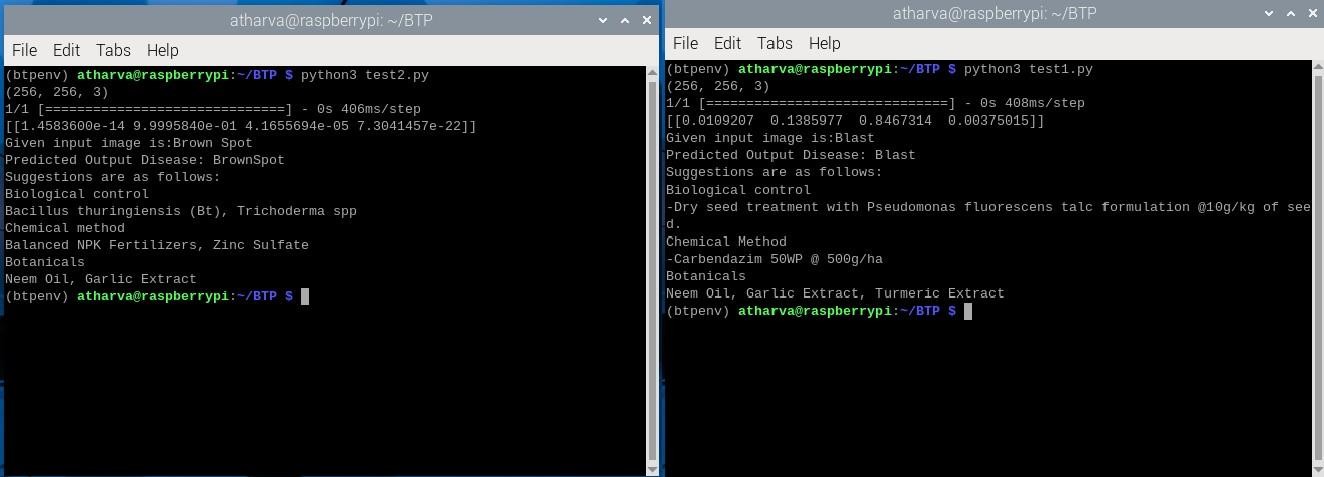
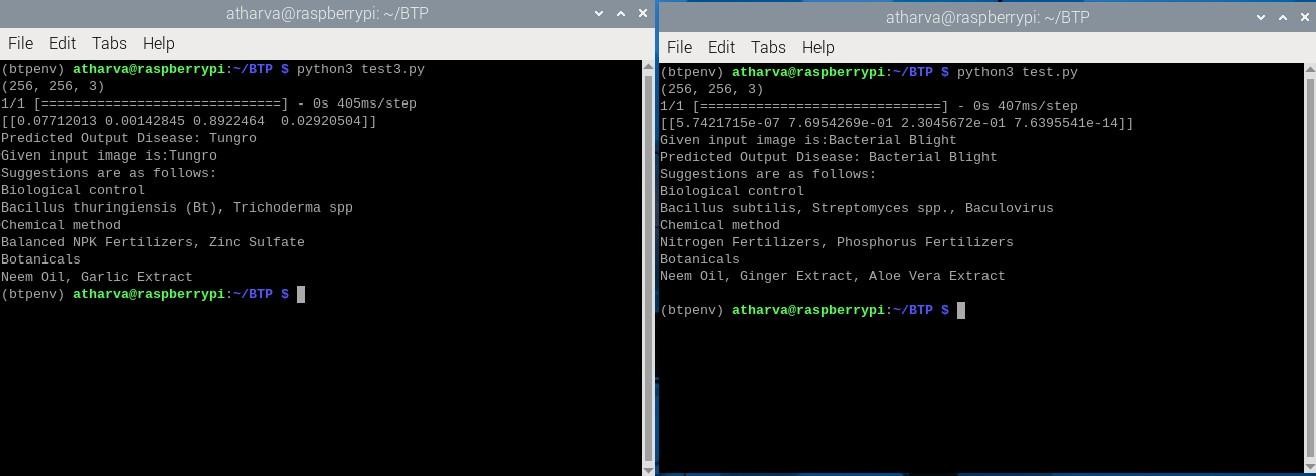


Fig.6.6. Raspberry Pi Output

With the script configured, it is executed directly on the Raspberry Pi through the Geaney editor. The code orchestrates the loading of the pre-trained model, processes incoming images of rice plants, and conducts inference to identify any indications of disease. The results of the detection process, such as classification labels or bounding box coordinates, are then presented on the Raspberry Pi terminal interface. This terminal output can be conveniently accessed remotely via RealVNC viewer, facilitating seamless real- time monitoring and diagnosis of rice diseases directly from the Raspberry Pi. This holistic setup offers an accessible and efficient solution for agricultural applications, empowering users with timely insights into crop health and facilitating informed decision-making.

CHAPTER 7

### Chapter 7 Advantages and Applications

This chapter delves into the advantages and various applications of the proposed flood prediction system, showcasing its potential impact across multiple sectors. In the section on advantages, the chapter highlights how the system surpasses traditional methods in terms of prediction accuracy, cost- effectiveness, simplicity of inference, integration capabilities, and tailored alerts. These advantages underscore the system's potential to revolutionize flood prediction and management practices. Subsequently, the chapter explores diverse applications of the technology, illustrating its relevance in disaster preparedness, agriculture, urban planning, and environmental conservation. Each application area demonstrates how the system can empower stakeholders to make informed decisions, mitigate risks, and enhance resilience in the face of flooding events.

#### Advantages:

##### Detection:

Utilizing deep learning models facilitates the early-stage detection of rice leaf diseases, allowing for prompt intervention and mitigating potential crop damage.

##### Precision agriculture:

Precision agriculture reduces the need for indiscriminate use of pesticides and resources by enabling farmers to precisely detect certain diseases, such as brown spot, leaf blight, leaf blast, and tungro. This precision farming method lessens its influence on the environment while promoting sustainable farming practices.

##### Real-time Decision help:

Farmers can now get real-time decision help in the field thanks to the availability of a web-based platform for disease prediction. This flexibility makes it possible to react quickly to new threats and changes in the health of crops.

##### Customized Treatment Recommendations:

The system can provide specific remedies tailored to the identified disease, optimizing the use of agrochemicals and minimizing unnecessary applications.

##### Raspberry Pi Integration for Real-World Application:

The integration of the system with Raspberry Pi facilitates real-world deployment, making it

accessible to farmers in diverse settings, including those in remote or resource-limited areas.

#### Applications:

##### Agricultural Disease Management:

Facilitates precise identification and targeted control of a wide range of rice leaf diseases, including but not limited to leaf blight, tungro, leaf blast, and brown spot, thereby enabling farmers to implement more effective disease management strategies in agriculture.

Empowers farmers with timely and accurate diagnosis of rice diseases, allowing for proactive measures such as disease-resistant crop selection, crop rotation, and optimal timing of pesticide application, ultimately minimizing crop losses and maximizing yields.

##### Educational Tool:

Serves as a comprehensive educational resource for agricultural students, researchers, and extension workers, offering valuable insights into the identification, diagnosis, and management of rice diseases through a sophisticated, technology-driven approach.

Provides access to a vast repository of annotated image datasets and expert recommendations, facilitating hands-on learning experiences and fostering a deeper understanding of the complexities of rice disease management.

##### Agrochemical Industry:

Facilitates data-driven decision-making processes within the agrochemical industry by offering valuable insights into disease prevalence, distribution, and trends derived from deep learning models.

Supports the development of targeted and sustainable solutions for rice diseases, including novel pesticides, biological control agents, and crop protection technologies, tailored to specific disease profiles and agricultural contexts.

##### Environment-Friendly Farming:

Contributes to the promotion of environmentally friendly farming practices by reducing the reliance on indiscriminate use of pesticides and fertilizers through precise disease diagnosis and targeted management strategies.

Supports the transition towards sustainable agriculture by minimizing chemical inputs, preserving soil health, and mitigating environmental pollution, thereby fostering long-term resilience and viability of agricultural ecosystems.

CHAPTER 8

### Chapter 8 Conclusion and Future Scope

In a comprehensive effort to address the challenges facing rice cultivation, we undertook a thorough investigation and analysis of diseases affecting rice plants, leveraging advanced techniques in deep learning. By deploying a tailored deep learning model, we achieved precise and reliable disease detection, enabling us to offer targeted remedial suggestions for each identified rice disease. Through rigorous evaluation, we confirmed the consistent and dependable performance of our deep learning model. To enhance practicality and accessibility, we implemented hardware solutions using Raspberry Pi, providing farmers with user-friendly tools for efficient crop management and proactive disease prevention. This integrated approach not only equips farmers with actionable insights but also empowers them to safeguard their crops effectively, contributing to sustainable agriculture practices and food security.

#### Conclusion:

In conclusion, our study effectively tackled the task of rice disease detection employing Convolutional Neural Networks (CNNs), targeting four major diseases: leaf blast, leaf blight, brown spot, and tungro. Through rigorous evaluation using metrics like accuracy, precision, recall, and F1 score, our model demonstrated outstanding performance, attaining an impressive **98%** overall accuracy with minimal loss (0.0669).

We also added further capability to our detection model by incorporating it into an easy-to-use web application that is created using Stream lit. Farmers can quickly diagnose sick rice leaves by uploading photos of the leaves to this portal. In addition, it provides treatment recommendations that may be put into practice and are divided into three categories: botanical pesticides, chemical pesticides, and biopesticides. With the use of these insights, farmers are better equipped to manage the effects of crop diseases by making well-informed decisions.

By combining cutting-edge machine learning with intuitive web tools, our method enhances the identification of rice diseases and promotes sustainable agricultural practices. By helping farmers maintain crop health and maximize harvests, this integrated system improves rice cultivation's resilience and productivity.

#### Future Scope:

Rice Leaf Disease Detection and Remedies Using Deep Learning" has bright future prospects that could lead to major breakthroughs in agriculture technology. Subsequent investigations can focus on improving deep learning models designed for rice crop disease detection. Larger datasets and the investigation of more complex neural network architectures can be used to attain improved accuracy, which would enable more accurate illness identification. The model's predictive power can be further enhanced by adding more data , such as environmental conditions, to give a more comprehensive picture of the dynamics of disease in rice fields.

The efficacy of the remedy recommendation system can be greatly increased by supplementing it with real- time data and regular updates based on feedback and continuing research. With the help of this dynamic method, farmers are better equipped to effectively treat rice leaf diseases by receiving timely and relevant recommendations that are suited to their unique circumstances. Furthermore, new developments in technology, such the use of drones and Internet of Things devices for better surveillance, present exciting opportunities to transform rice farming's approaches to managing disease. For these innovations to be widely adopted, efforts must be made to improve decision support tools and mobile applications' usability and accessibility for farmers and stakeholders. Researcher-farmer and industrial partner collaboration will continue to be crucial to the practical applicability and successful deployment of these technologies in rice crop management.

CHAPTER 9

### Chapter 9 References

1. Dr.Rajani P.K,Dr.Vaidehi V Deshmukh,Dr.Sheetal U. Bhandari,Dr.Roshani Raut, Dr.Reena Kharat, “Rice Leaf Disease Detection Using Convolutional Neural Network", International Journal on Recent and Innovation Trends in Computing and Communication (IJRITCC) Published by Auricle Global Society of Education and Research, Vol. 11, Issue no.10s, pp. 512–517, 7th October 2023 ISSN (Online): 2321-8169
2. Guruprasad Deshpande,Rajani P.K, Vishal Khandagle,Jayesh Kolhe, “Comparison of Classification Algorithm for Crop Decision based on Environmental Factors using Machine Learning ", International Journal on Recent and Innovation Trends in Computing and Communication (IJRITCC) Published by Auricle Global Society of Education and Research, Vol. 11, Issue no.9s, pp. 360–368,31st August 2023(SCOPUS-Q4) ISSN (Online): 2321-8169
3. Dr.Rajani P.K, Kalyani Patil, Bhagyashree Marathe, Prerna Mhaisane, Atharva Tundalwar, “Heart Disease Prediction using different Machine Learning Algorithms", International Journal on Recent and Innovation Trends in Computing and Communication (IJRITCC) Published by Auricle Global Society of Education and Research, Vol. 11, Issue no.9s, pp. 354–359,31st August 2023(SCOPUS-Q4) ISSN (Online): 2321-8169
4. Shruti Aggarwal ,M. Suchithra, N. Chandramouli, Macha Sarada, Amit Verma, D. Vetri Thangam, Bhaskar Pant, and Biruk Ambachew Adugna, “Rice Disease Detection Using Artificial Intelligence and Machine Learning Techniques to Improvise Agro-Business ,”Hindawi

Scientific Programming Volume 2022.

1. B. S. Bari, M. N. Islam, M. Rashid, M. J. Hasan, M. A. M. Razman, R. M. Musa, A. F. A. Nasir, and

A. P. Majeed, ‘‘A real-time approach of diagnosing Rice leaf disease using deep learning-based faster R-CNN framework,’’ PeerJ Comput. Sci., vol. 7, pp. 1–27, Apr. 2021.

1. N. P. S. Rathore and L. Prasad, “Automatic rice plant disease recognition and identification using convolutional neural network,” 2018 Journal of Critical Reviews, vol.7(15) pp. 6076–6086, 2020.
2. Xiaoyue Xie, Yuan Ma, Bin Liu, Jinrong He, Shuqin Li, and HongyanWang. A deep-learning based real-time detector for grape leaf diseases using improved convolutional neural networks. Frontiers in plant science, 11:751, 2020.
3. D. Li, R. Wang, C. Xie et al., “A recognition method for rice plant diseases and pests video detection based on deep convolutional neural network,” Sensors, vol. 20, no. 3, p. 578, 2020.
4. Jun Sun, Yu Yang, Xiaofei He, and Xiaohong Wu. Northern maize leaf blight detection under a complex field environment based on deep learning. IEEE Access, 8:33679–33688, 2020.
5. M. E. Pothen and M. L. Pai, “Detection of rice leaf diseases using image processing,” 2020 Fourth

International Conference on Computing Methodologies and Communication (ICCMC),pp. 424–430, 2020.

[11] K. Ahmed, T. R. Shahidi, S. M. I. Alam and S. Momen “Rice leaf disease detection using machine learning techniques,” 2019 International Conference on Sustainable Technologies for Industry 4.0 (STI),pp. 1-5, 2019.

[12] S. Umadevi and K. S. J. Marseline, “A survey on data mining classification algorithms,” Proc. IEEE Int. Conf. Signal Process. Communication. ICSPC 2017, vol. 2018-January, no. July, pp. 264–268, 2018, doi: 10.1109/CSPC.2017.8305851.

1. R. R. Atole and D. Park, “A multiclass deep convolutional neural network classifier for detection of common rice plant anomalies,” International Journal of Advanced Computer Science and Applications, vol. 9, pp. 67–70, 2018.
2. T. Islam, M. Sah, S. Baral and R. RoyChoudhury, “A faster technique on rice disease detection using image processing of affected area in agrofield, ” 2018 Second International Conference on Inventive Communication and Computational Technologies (ICICCT), pp. 62-66, 2018.
3. F. T. Pinki, N. Khatun, and S. M. M. Islam, “Content based paddy leaf disease recognition and remedy prediction using support vector machine,” in Proceedings of the 20th International Conference on Information Technology, Dhaka, Bangladesh, January 2018.





1. **Raspberry Pi Model 4**











# Colophon

© 2020 Raspberry Pi (Trading) Ltd.

The documentation around the Raspberry Pi Compute Module 4 is licensed under a Creative Commons [Attribution-](https://creativecommons.org/licenses/by-nd/4.0/) [NoDerivatives 4.0 International](https://creativecommons.org/licenses/by-nd/4.0/) (CC BY-ND).

build-date: 2021-01-28

build-version: githash: 80b6c48-clean

##### Legal Disclaimeí Notice

TECHNICAL AND RELIABILITY DATA FOR RASPBERRY PI PRODUCTS (INCLUDING DATASHEETS) AS MODIFIED FROM TIME TO TIME (“RESOURCES”) ARE PROVIDED BY RASPBERRY PI (TRADING) LTD (“RPTL) "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW IN NO EVENT SHALL RPTL BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THE RESOURCES, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

RPTL reserves the right to make any enhancements, improvements, corrections or any other modifications to the RESOURCES or any products described in them at any time and without further notice.

The RESOURCES are intended for skilled users with suitable levels of design knowledge. Users are solely responsible for their selection and use of the RESOURCES and any application of the products described in them. User agrees to indemnify and hold RPTL harmless against all liabilities, costs, damages or other losses arising out of their use of the RESOURCES.

RPTL grants users permission to use the RESOURCES solely in conjunction with the Raspberry Pi products. All other use of the RESOURCES is prohibited. No licence is granted to any other RPTL or other third party intellectual property right.

HIGH RISK ACTIVITIES. Raspberry Pi products are not designed, manufactured or intended for use in hazardous environments requiring fail safe performance, such as in the operation of nuclear facilities, aircraft navigation or communication systems, air traffic control, weapons systems or safety-critical applications (including life support systems and other medical devices), in which the failure of the products could lead directly to death, personal injury or severe physical or environmental damage (“High Risk Activities”). RPTL specifically disclaims any express or implied warranty of fitness for High Risk Activities and accepts no liability for use or inclusions of Raspberry Pi products in High Risk Activities.

Raspberry Pi products are provided subject to RPTL’s [Standard Terms.](https://www.raspberrypi.org/terms-conditions-sale/) RPTL’s provision of the RESOURCES does not expand or otherwise modify RPTL’s [Standard Terms](https://www.raspberrypi.org/terms-conditions-sale/) including but not limited to the disclaimers and warranties expressed in them.

# Table of Contents

[Colophon](#_bookmark0) [1](#_bookmark0)

[Legal Disclaimer Notice](#_bookmark1) [1](#_bookmark1)

* 1. [Introduction](#_bookmark2) [3](#_bookmark2)
     1. [Introduction](#_bookmark3) [3](#_bookmark3)
     2. [Features](#_bookmark5) [3](#_bookmark5)
  2. [Interfaces](#_bookmark6) [5](#_bookmark6)
     1. [Wireless](#_bookmark7) [5](#_bookmark7)
        1. [WL\_nDisable](#_bookmark8) [5](#_bookmark8)
        2. [BT\_nDisable](#_bookmark9) [5](#_bookmark9)
     2. [Ethernet](#_bookmark10) [6](#_bookmark10)
     3. [PCIe (Gen2 x1)](#_bookmark12) [6](#_bookmark12)
     4. [USB 2.0 (Highspeed)](#_bookmark13) [7](#_bookmark13)
     5. [GPIO](#_bookmark14) [7](#_bookmark14)
        1. [Alternative Function Assignments](#_bookmark15) [8](#_bookmark15)
     6. [Dual HDMI 2.0](#_bookmark16) [10](#_bookmark16)
     7. [CSI-2 (MIPI Serial Camera)](#_bookmark17) [10](#_bookmark17)
     8. [DSI (MIPI Serial Display)](#_bookmark18) [11](#_bookmark18)
     9. [I2C (SDA0 SCL0)](#_bookmark19) [11](#_bookmark19)
     10. [I2C (ID\_SD ID\_SC)](#_bookmark20) [11](#_bookmark20)
     11. [SDIO/eMMC (CM4Lite only)](#_bookmark21) [11](#_bookmark21)
     12. [Analog IP0/IP1](#_bookmark22) [12](#_bookmark22)
     13. [Global\_EN](#_bookmark23) [12](#_bookmark23)
     14. [RUN\_PG](#_bookmark24) [12](#_bookmark24)
     15. [nRPI\_BOOT](#_bookmark25) [12](#_bookmark25)
     16. [LED\_nACT](#_bookmark26) [12](#_bookmark26)
     17. [LED\_nPWR](#_bookmark27) [13](#_bookmark27)
     18. [EEPROM\_nWP](#_bookmark28) [13](#_bookmark28)
  3. [Electrical and Mechanical](#_bookmark29) [14](#_bookmark29)
     1. [Mechanical](#_bookmark30) [14](#_bookmark30)
     2. [Thermal](#_bookmark31) [15](#_bookmark31)
     3. [Electrical Specification](#_bookmark32) [15](#_bookmark32)
  4. [Pin Out](#_bookmark35) [17](#_bookmark35)
     1. [Differential Pairs](#_bookmark36) [23](#_bookmark36)

4.1.1. [100Ω Differential pairs signal lengths](#_bookmark37) [23](#_bookmark37)

4.1.2. [90Ω Differential Pairs signal lengths](#_bookmark39) [25](#_bookmark39)

* 1. [Power](#_bookmark41) [27](#_bookmark41)
     1. [Power up sequencing](#_bookmark42) [27](#_bookmark42)
     2. [Power down sequencing](#_bookmark43) [27](#_bookmark43)
     3. [Power Consumption](#_bookmark44) [27](#_bookmark44)
     4. [Regulator Outputs](#_bookmark45) [27](#_bookmark45)

[Appendix A: Troubleshooting](#_bookmark46) [28](#_bookmark46)

[Hardware Checklist](#_bookmark47) [28](#_bookmark47)

[Bootloader](#_bookmark48) [28](#_bookmark48)

[rpi-eeprom-update](#_bookmark49) [28](#_bookmark49)

[EEPROM Write protect](#_bookmark50) [28](#_bookmark50)

[Firmware](#_bookmark51) [29](#_bookmark51)

[Kernel](#_bookmark52) [29](#_bookmark52)

[Appendix B: Availability](#_bookmark53) [30](#_bookmark53)

[Support](#_bookmark54) [30](#_bookmark54)

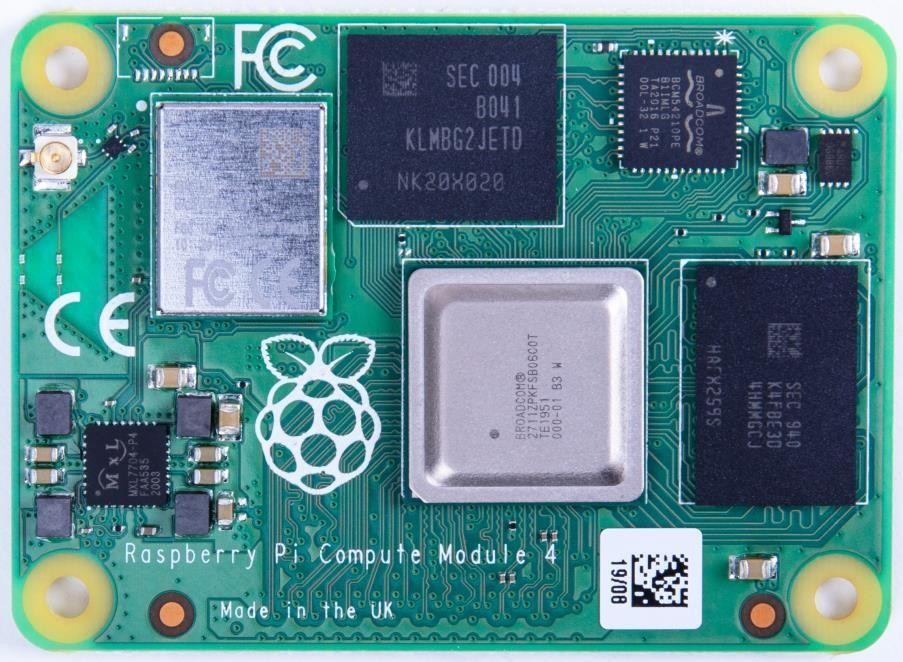
[Ordering codes](#_bookmark55) [30](#_bookmark55)

[Packaging](#_bookmark56) [31](#_bookmark56)

**Chapter 1. Introduction**

##### Intíoduction

*Figuíe 1. ľhe Raspbeííy Pi Compute Module 4 (CM4).*



The Raspberry Pi Compute Module 4 (CM4) is a System on Module (SoM) containing processor, memory, eMMC Flash and supporting power circuitry. These modules allow a designer to leverage the Raspberry Pi hardware and software stack in their own custom systems and form factors. In addition these modules have extra IO interfaces over and above what is available on the Raspberry Pi boards, opening up more options for the designer.

The design of the CM4 is loosely based on the Raspberry Pi 4, Model B, and for cost sensitive applications it can be supplied without the eMMC fitted; this version is called the Raspberry Pi Compute Module 4 Lite (CM4Lite).

While [previous generations of the Compute Module](https://www.raspberrypi.org/documentation/hardware/computemodule/datasheet.md) have all shared the same DDR2-SODIMM-mechanically-compatible form factor, the new CM4 and CM4Lite are different. The electrical interface of the CM4 is via two 100 -pin high density connectors, and the new physical form factor has a smaller footprint overall when the connectors are taken into account.

This change is due to the addition of new interfaces; an additional second HDMI, PCIe, and Ethernet. The addition of these new interfaces, especially PCIe, would not have been possible while preserving the previous form factor.

 **NOľE**

Unless otherwise stated, for this document CM4 also refers to CM4Lite.

##### Ïeatuíes

Key features of the CM4 are as follows:

* Broadcom [BCM2711,](https://datasheets.raspberrypi.org/bcm2711/bcm2711-peripherals.pdf) Quad core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz
* Small Footprint 55mm × 40mm × 4.7mm module
  + 4 × M2.5 mounting holes
* H.265 (HEVC) (upto 4Kp60 decode), H.264 (upto 1080p60 decode, 1080p30 encode)
* OpenGL ES 3.0 graphics
* Options for 1GB, 2GB, 4GB or 8GB LPDDR4-3200 SDRAM with ECC (see [Appendix B)](#_bookmark53)
* Options for 0GB **(CM4Lite)**, 8GB, 16GB, or 32GB eMMC Flash memory (see [Appendix B)](#_bookmark53)
  + Peak eMMC bandwidth 100MBytes/s (four times faster than previous Compute modules)
* Option (see [Appendix B) f](#_bookmark53)or certified radio module with:
  + 2.4 GHz, 5.0 GHz IEEE 802.11 b/g/n/ac wireless
  + Bluetooth 5.0, BLE
  + On board electronic switch to select between PCB trace or external antenna
* Gigabit Ethernet PHY supporting IEEE 1588
* 1 × PCIe 1-lane Host, Gen 2 ( 5Gbps )
* 1 × USB 2.0 port ( highspeed )
* 28 × GPIO supporting either 1.8v or 3.3v signalling and peripheral options:
  + Up to 5 × UART
  + Up to 5 × I2C
  + Up to 5 × SPI
  + 1 × SDIO interface
  + 1 × DPI (Parallel RGB Display)
  + 1 × PCM
  + Up to 2× PWM channels
  + Up to 3× GPCLK outputs
* 2 × HDMI 2.0 ports (up to 4Kp60 supported)
* MIPI DSI:
  + 1 × 2-lane MIPI DSI display port
  + 1 × 4-lane MIPI DSI display port
* MIPI CSI-2:
  + 1 × 2-lane MIPI CSI camera port
  + 1 × 4-lane MIPI CSI camera port
* 1 × SDIO 2.0 **(CM4Lite)**
* Single +5v PSU input.

# Chapter 2. Interfaces

##### Wiíeless

The CM4 can be supplied with an onboard wireless module based on the Broadcom BCM43455 supporting both,

* + - 2.4 GHz, 5.0 GHz IEEE 802.11 b/g/n/ac wireless
    - Bluetooth 5.0, BLE.

These wireless interfaces can be individually enabled or disabled as required. For instance, in the case of a kiosk

application, a service engineer could enable wireless operation and then disable it once finished.

The CM4 has an onboard antenna. If used it should be positioned in the product such that it is not surrounded by metal, including any ground plane (see [Chapter 3 f](#_bookmark29)or further details). Alternatively there is a standard U.FL connector on the module, see [Figure 1,](#_bookmark4) so that an external antenna can be used.

Raspberry Pi has an antenna kit which is certified to be used with the CM4. If a different antenna is used then separate certification will be required.

 **WARNING**

Raspberry Pi Trading will not be able to assist with certification for third-party antennas.

The selection of internal or external antenna is done at boot time using the config.txt file, and can not be changed during operation. The config.txt options are dtparam=ant1 to select the internal antenna, or dtparam=ant2 for the external antenna.

* + 1. WL\_nDisable

This pin serves a number of functions;

* + - 1. It can be used to monitor the enable/disable state of wireless networking. A logic high means the wireless networking module is powered up.
      2. When driven or tied low it prevents the wireles network module from powering up. This is useful to reduce power consumption or in applications where it is required to physically ensure the wireless networking is disabled. If the interface is enabled after being disabled, the wireless interface driver needs reinitalised.

 **NOľE**

On CM4 modules without wireless, this pin is reserved.

* + 1. BT\_nDisable

This pin serves a number of functions;

* + - 1. It can be used to monitor the enable/disable state of Bluetooth. A logic high means the Bluetooth module is powered up.
      2. When driven, or tied low, it prevents the Bluetooth module from powering up. This is useful to reduce power consumption, or in applications where it is required to physically ensure the Bluetooth is disabled. If the interface is enabled after being disabled, the Bluetooth interface driver needs reinitalised.

 **NOľE**

On CM4 modules without wireless, this pin is reserved.

*Figuíe 2. Etheínet schematic inteíface foí the Raspbeííy Pi Compute Module 4 suppoíting POE, and with added ESD píotection.*

##### Etheínet

The CM4 has an onboard Gigabit Ethernet PHY — the Broadcom [BCM54210PE](https://www.broadcom.com/products/ethernet-connectivity/phy-and-poe/copper/gigabit/bcm54210) — some of the major features of this PHY include;

* [IEEE 1588-2008](https://standards.ieee.org/standard/1588-2008.html) compliant
* Detection and correction of swapped pairs
* MDI crossover, pair skew and pair polarity correction

A standard 1:1 RJ45 MagJack is all that is necessary to provide an Ethernet connection to the CM4. Typical wiring of a

MagJack supporting POE, and with added ESD protection, can be seen in [Figure 2.](#_bookmark11)



The differential Ethernet signals should be routed as 100Ω differential pairs, with suitable clearances. Length matching between pairs should be better than 50mm, so in the typical case no length matching is required. However the signals within a pair need to be length matched, ideally to better than 0.15mm.

The PHY also supports up to 3 LEDs to give user status feedback, these are low active. These LEDs can have a range of functions, and you should consult your OS driver to see which functions are supported by your driver.

The PHY also provides SYNC\_IN and SYNC\_OUT at 1.8v signalling to support [IEEE 1588-2008.](https://standards.ieee.org/standard/1588-2008.html)

##### PCIe (Gen2 x1)

The CM4 has an internal PCIe 2.0 x1 host controller. While on the Raspberry Pi 4, Model B this has been connected to a USB 3 host controller (using the Via Labs [VLI805),](https://www.via-labs.com/product_show.php?id=48) on the CM4 the product designer is free to choose how the interface is used.

 **WARNING**

You should ensure that there is a suitable OS driver for any host controller that is chosen before proceeding to a prototype.

 **NOľE**

The onboard PCIe Host controller doesn’t support 64bit accesses from the ARM, they must be split up into two 32 bit accesses.

Connecting a PCIe device follows the standard PCIe convention. The CM4 has onboard AC coupling capacitors for CLK and PCIe\_TX signals. However the PCIe\_RX signals need external coupling capacitors close to the driving source (the device TX), if you are using an external PCIe/NVMe cards these capacitors will be onboard. The PCIe conversion is that if

you are wiring directly to an IC then the TX and RX pairs need to be swapped ( ie. TX->RX , Rx->TX ). If you are wiring to a connector then this is typically labeled from the host post of view and so TX RX swaps aren’t required. Additionally the PCIe\_CLK\_nREQ must be connected to ensure the CM4 produces a clock signal, and the PCIe\_nRST should also be connected to ensure the device is correctly reset when required.

The differential PCIe signals should be routed as 90Ω differential pairs, with suitable clearances. There is no need to match the lengths between pairs, only the signals within a Pair need to be length matched ideally to better than 0.1mm.

 **ľIP**

5.10 kernels and newer have had support for MSI-X added. There is a limit of upto 32 IRQs available. If the device has problems with interrupts then adding pci=nomsi to cmdline.txt (and rebooting) often fixes the issue.

##### USB 2.0 (Highspeed)

The USB 2.0 interface supports up to 480MBps signalling. The differential pair should be routed as a 90Ω differential pair. The P N signals should ideally be matched to 0.15mm

 **ľIP**

The USB interface is disabled to save power by default on the CM4 . To enable it you need to add

dtoverlay=dwc2,dr\_mode=host to the config.txt file

 **NOľE**

The port is capable of being used as a true USB On-The-Go (OTG) port. While there is no official documentation, some users have had success making this work. The USB\_OTG pin is used to select between USB host and device that is typically wired to the ID pin of a Micro usb connector. To use this functionality it must be enabled in the OS that is used. If using either as a fixed slave or fixed master, please tie the USB OTGID pin to ground

##### GPIO

There are 28 pins available for general purpose I/O (GPIO), which correspond to the GPIO pins on the Raspberry Pi 4, Model B 40-pin header. These pins have access to internal peripherals; I2C, PWM, SPI, and UART. The [BCM2711](https://datasheets.raspberrypi.org/bcm2711/bcm2711-peripherals.pdf) ARM Peripherals book describes these features in detail, and the multiplexing options available. The drive strength and slew rate should ideally be set as low as possible to reduce any EMC issues. GPIO2 and GPIO3 have 1.8K pull up resistors.

The BCM2711 GPIO bank is powered by GPIO\_VREF, this can either be connected to the +1.8v from the CM4 for 1.8v signalling GPIO, or +3.3v from the CM4 for +3.3v signalling. You should keep the load on the 28 GPIO pins to below

50mA

in total. GPIO\_VREF must be powered for the CM4 to startup correctly

**2.5.1. Alteínative Ïunction Assignments**

*ľable 1. GPIO Pins Alteínative Function Assignment*

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **GPIO** | **Pull** | **ALľ0** | **ALľ1** | **ALľ2** | **ALľ3** | **ALľ4** | **ALľ5** |
| GPIO0 | High | SDA0 | SA5 | PCLK | SPI3\_CE0\_N | TXD2 | SDA6 |
| GPIO1 | High | SCL0 | SA4 | DE | SPI3\_MISO | RXD2 | SCL6 |
| GPIO2 | High | SDA1 | SA3 | LCD\_VSYNC | SPI3\_MOSI | CTS2 | SDA3 |
| GPIO3 | High | SCL1 | SA2 | LCD\_HSYNC | SPI3\_SCLK | RTS2 | SCL3 |
| GPIO4 | High | GPCLK0 | SA1 | DPI\_D0 | SPI4\_CE0\_N | TXD3 | SDA3 |
| GPIO5 | High | GPCLK1 | SA0 | DPI\_D1 | SPI4\_MISO | RXD3 | SCL3 |
| GPIO6 | High | GPCLK2 | SOE\_N / SE | DPI\_D2 | SPI4\_MOSI | CTS3 | SDA4 |
| GPIO7 | High | SPI0\_CE1\_N | SWE\_N  / SRW\_N | DPI\_D3 | SPI4\_SCLK | RTS3 | SCL4 |
| GPIO8 | High | SPI0\_CE0\_N | SD0 | DPI\_D4 | BSCSL / CE\_N | TXD4 | SDA4 |
| GPIO9 | Low | SPI0\_MISO | SD1 | DPI\_D5 | BSCSL / MISO | RXD4 | SCL4 |
| GPIO10 | Low | SPI0\_MOSI | SD2 | DPI\_D6 | BSCSL SDA  / MOSI | CTS4 | SDA5 |
| GPIO11 | Low | SPI0\_SCLK | SD3 | DPI\_D7 | BSCSL SCL  / SCLK | RTS4 | SCL5 |
| GPIO12 | Low | PWM0\_0 | SD4 | DPI\_D8 | SPI5\_CE0\_N | TXD5 | SDA5 |
| GPIO13 | Low | PWM0\_1 | SD5 | DPI\_D9 | SPI5\_MISO | RXD5 | SCL5 |
| GPIO14 | Low | TXD0 | SD6 | DPI\_D10 | SPI5\_MOSI | CTS5 | TXD1 |
| GPIO15 | Low | RXD0 | SD7 | DPI\_D11 | SPI5\_SCLK | RTS5 | RXD1 |
| GPIO16 | Low | <reserved> | SD8 | DPI\_D12 | CTS0 | SPI1\_CE2\_N | CTS1 |
| GPIO17 | Low | <reserved> | SD9 | DPI\_D13 | RTS0 | SPI1\_CE1\_N | RTS1 |
| GPIO18 | Low | PCM\_CLK | SD10 | DPI\_D14 | SPI6\_CE0\_N | SPI1\_CE0\_N | PWM0\_0 |
| GPIO19 | Low | PCM\_FS | SD11 | DPI\_D15 | SPI6\_MISO | SPI1\_MISO | PWM0\_1 |
| GPIO20 | Low | PCM\_DIN | SD12 | DPI\_D16 | SPI6\_MOSI | SPI1\_MOSI | GPCLK0 |
| GPIO21 | Low | PCM\_DOUT | SD13 | DPI\_D17 | SPI6\_SCLK | SPI1\_SCLK | GPCLK1 |
| GPIO22 | Low | SD0\_CLK | SD14 | DPI\_D18 | SD1\_CLK | ARM\_TRST | SDA6 |
| GPIO23 | Low | SD0\_CMD | SD15 | DPI\_D19 | SD1\_CMD | ARM\_RTCK | SCL6 |
| GPIO24 | Low | SD0\_DAT0 | SD16 | DPI\_D20 | SD1\_DAT0 | ARM\_TDO | SPI3\_CE1\_N |
| GPIO25 | Low | SD0\_DAT1 | SD17 | DPI\_D21 | SD1\_DAT1 | ARM\_TCK | SPI4\_CE1\_N |
| GPIO26 | Low | SD0\_DAT2 | <reserved> | DPI\_D22 | SD1\_DAT2 | ARM\_TDI | SPI5\_CE1\_N |
| GPIO27 | Low | SD0\_DAT3 | <reserved> | DPI\_D23 | SD1\_DAT3 | ARM\_TMS | SPI6\_CE1\_N |
| GPIO44 | - | GPCLK1 | SDA0 | SDA1 | <reserved> | SPI0\_CE1\_N | SD\_CARD\_V OL T |

Up to 6 alternate functions are available. The [BCM2711](https://datasheets.raspberrypi.org/bcm2711/bcm2711-peripherals.pdf) ARM Peripherals book describes these features in detail. The table below gives a quick overview.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **GPIO** | **Pull** | **ALľ0** | **ALľ1** | **ALľ2** | **ALľ3** | **ALľ4** | **ALľ5** |
| GPIO45 | - | PWM0\_1 | SCL0 | SCL1 | <reserved> | SPI0\_CE2\_N | SD\_CARD\_P W R0 |

Special function legend:

*ľable 2. GPIO Pins Alteínative Function Legend*

|  |  |
| --- | --- |
| **Name** | **Ïunction** |
| SDA0 | BSC master 0 data linea |
| SCL0 | BSC master 0 clock line |
| SDAx | BSC master 1,3,4,5,6 data lineb |
| SCLx | BSC master 1,3,4,5,6 clock line |
| GPCLKx | General purpose Clock 0,1,2 |
| SPIx\_CE2\_N | SPI 0,3,4,5,6 Chip select 2 |
| SPIx\_CE1\_N | SPI 0,3,4,5,6 Chip select 1 |
| SPIx\_CE0\_N | SPI 0,3,4,5,6 Chip select 0 |
| SPIx\_MISO | SPI 0,3,4,5,6 MISO |
| SPIx\_MOSI | SPI 0,3,4,5,6 MOSI |
| SPIx\_SCLK | SPI 0,3,4,5,6 Serial clock |
| PWMx\_0 | PWM 0,1 channel 0 |
| PWMx\_1 | PWM 0,1 channel 1 |
| TXDx | UART 0,2,3,4,5 Transmit Data |
| RXDx | UART 0,2,3,4,5 Receive Data |
| CTSx | UART 0,2,3,4,5 Clear To Send |
| RTSx | UART 0,2,3,4,5 Request To Send |
| PCM\_CLK | PCM clock |
| PCM\_FS | PCM Frame Sync |
| PCM\_DIN | PCM Data in |
| PCM\_DOUT | PCM data out |
| SAx | Secondary mem Address bus |
| SOE\_N / SE | Secondary mem. Controls |
| SWE\_N / SRW\_N | Secondary mem. Controls |
| SDx | Secondary mem. data bus |
| BSCSL SDA / MOSI | BSC slave Data, SPI slave MOSI |
| BSCSL SCL / SCLK | BSC slave Clock, SPI slave clock |
| BSCSL - / MISO | BSC <not used>, SPI MISO |
| BSCSL - / CE\_N | BSC <not used>, SPI CSn |
| SPI1\_CE2\_N | SPI 1 Chip select 2 c |
| SPI1\_CE1\_N | SPI 1 Chip select 1 |
| SPI1\_CE0\_N | SPI 1 Chip select 0 |

|  |  |
| --- | --- |
| **Name** | **Ïunction** |
| SPI1\_MISO | SPI 1 MISO |
| SPI1\_MOSI | SPI 1 MOSI |
| SPI1\_SCLK | SPI 1 Serial clock |
| TXD1 | UART 1 Transmit Data |
| RXD1 | UART 1 Receive Data |
| CTS1 | UART 1 Clear To Send |
| RTS1 | UART 1 Request To Send |
| ARM\_TRST | ARM JTAG reset |
| ARM\_RTCK | ARM JTAG return clock |
| ARM\_TDO | ARM JTAG Data out |
| ARM\_TCK | ARM JTAG Clock |
| ARM\_TDI | ARM JTAG Data in |
| ARM\_TMS | ARM JTAG Mode select |
| PCLK | Display Parallel Interface |
| DE | Display Parallel Interface |
| LCD\_VSYNC | Display Parallel Interface |
| LCD\_HSYNC | Display Parallel Interface |
| DPI\_Dx | Display Parallel Interface |

a The Broadcom Serial Control bus is a proprietary bus compliant with the Philips® I2C bus/interface.

b BSC master 2 & 7 are not user-accessible.

c SPI 2 is not user-accessible.

##### Dual HDMI 2.0

The CM4 supports two HDMI 2.0 interfaces each one capable of driving 4K images. If both HDMI outputs are used then each can be driven upto 4Kp30, however if only HDMI0 interface is being used then images up to 4Kp60 are possible.

HDMI signals should be routed as 100Ω differential pairs, each signal within a pair should ideally be matched to better than 0.15mm. Pairs don’t typically need any extra matching as they only have to be matched to 25mm.

CEC is also supported, an internal 27K pullup resistor is included in the CM4.

Basic onboard ESD protection is provided for the I2C EDID signals and the CEC signals, internal pullup and down resistors are also provided. On the {rpi4} the HDMI signals don’t have any extra ESD protection , depending on the application extra ESD protection maybe required.

##### CSI-2 (MIPI Seíial Cameía)

The CM4 supports two camera ports; CAM0 (2 lanes) and CAM1 (4 lanes). CSI signals should be routed as 100Ω differential pairs, each signal within a pair should ideally be matched to better than 0.15mm.

The documentation around the CSI interface can be found on the [Raspberry Pi website](https://www.raspberrypi.org/documentation/linux/software/libcamera/csi-2-usage.md) while [Linux kernel drivers](https://github.com/raspberrypi/linux/blob/rpi-5.4.y/drivers/media/platform/bcm2835/bcm2835-unicam.c) can be found on Github.

 **NOľE**

Camera sensors supported by the official Raspberry Pi firmware are; the OmniVision OV5647, Sony IMX219 and Sony IMX477, no security device is required on Compute Module devices to use these camera sensors.

##### DSI (MIPI Seíial Display)

The CM4 supports two display ports; DISP0 (2 lanes) and DISP1 (4 lanes). Each lane supports a maximum of data rate per lane of 1Gbit/s.

Although [Linux kernel drivers](https://github.com/raspberrypi/linux/blob/rpi-5.4.y/drivers/gpu/drm/vc4/vc4_dsi.c) are available, the DSI interface is not currently documented. Only DSI displays supported by the official Raspberry Pi firmware are supported, DSI signals should be routed as 100Ω differential pairs, each signal within a pair should ideally be matched to better than 0.15mm.

 **NOľE**

While only official DSI displays are supported, other displays can be added using the parallel DPI interface which is

available as a GPIO alternate function. The CM4 supports up to 3 displays of any type (HDMI, DSI, DPI) at any one time.

##### I2C (SDA0 SCL0)

This internal I2C bus is normally allocated to the CSI1 and DSI1 as these devices are controlled by the firmware. It can be used as a general I2C bus if the CSI1 ad DSI1 interfaces aren’t being used or are being controlled by the firmware. For example libcamera runs on the ARM and doesn’t use the firmware so in this case you may use CSI1 and this I2C bus. SDA0 is connected to GPIO44 on the BCM2711 and SCL0 is connected to GPIO45

##### I2C (ID\_SD ID\_SC)

This I2C bus is normally used for identifying HATs and controlling CSI0 and DSI0 devices. If the firmware isn’t using the I2C bus e.g. CSI0 and DSI0 aren’t being used then these pins may be used as GPIO 0 and GPIO 1 if required.

Note: If these pins are used as GPIO pins then to prevent the firmware from checking to see if there is a HAT EEPROM available add force\_eeprom\_read=0 to the config.txt file.

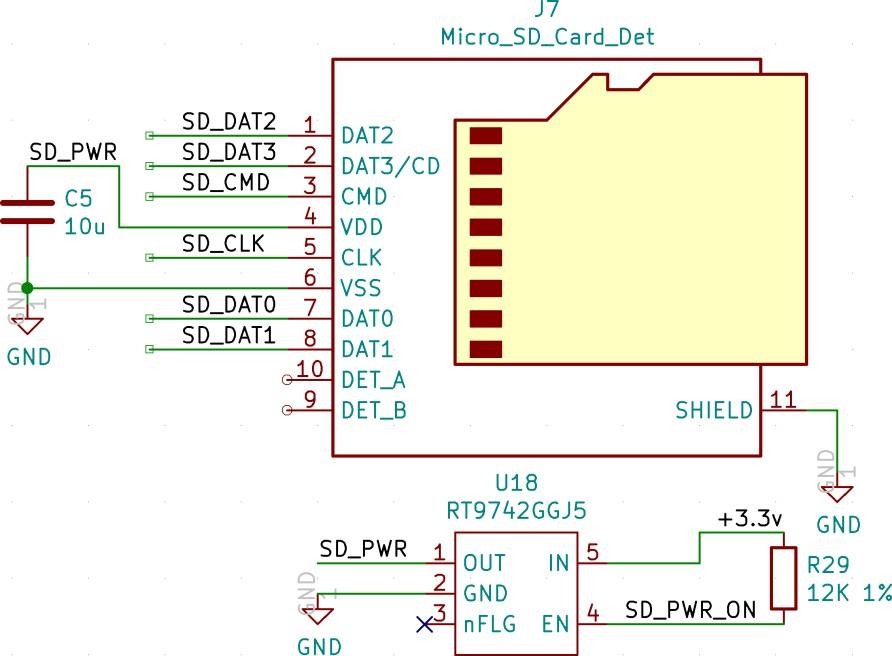
##### SDIO/eMMC (CM4Lite only)

If the CM4Lite is used, which does not have on-board eMMC, then the eMMC signals are available on the connector so that an external eMMC or SDCARD can be used.

The SD\_PWR\_ON signal is used to enable an external power switch to turn on power to the SDCARD, for eMMC it typically isn’t used. If booting from SDCARD is required then a pullup resistor must also be fitted to default the power to be on. When SD\_VDD\_override is high, this signal is used to force 1.8v signalling on the SDIO interface. Typically this is used with eMMC memory.

*Figuíe 3. CM4Lite SDCARD*

*inteíface.*



##### Analog IP0/IP1

These are the two spare inputs on the [MXL7704.](https://www.maxlinear.com/ds/mxl7704.pdf) The MXL7704 datasheet should be consulted if these pins are to be used. Onboard filtering is provided by a 100nF capacitor to ground for each signal. On the Raspberry Pi 4, Model B these are connected to the USB C connector CC1 and CC2 pins.

* 1. Global\_EN

Pulling this pin low puts the CM4 in the lowest possible power down state. After software shutdown Global\_EN needs to be pulled low for > 1ms to restart the power system on the CM4.

 **ľIP**

It is recommended to only pull this pin low once the OS has shutdown.

* 1. RUN\_PG

This pin when high signals that the CM4 has started. Driving this pin low resets the module, this should be done with caution as if files on a filesystem are open they will not be closed.

* 1. nRPI\_BOOT

During boot if this pin is low booting from eMMC will be stopped and booting will be transferred to rpi boot which is via USB.

* 1. LED\_nACT

This pin is designed to drive an LED to replicate the green LED on the Raspberry Pi 4, Model B. Under Linux this pin will flash to signify eMMC access, while if there is an error during booting this LED will flash error patterns which can be

decoded using the [look up table](https://www.raspberrypi.org/documentation/configuration/led_blink_warnings.md) on the Raspberry Pi website.

* 1. LED\_nPWR

This pin needs to be buffered to drive an LED. The signal is designed to replicate the red power LED on the Raspberry Pi 4, Model B.

* 1. EEPROM\_nWP

It is recommended that final products pull this pin low to prevent the end users changing the contents of the on board EEPROM. See the Raspberry Pi 4, Model B documentation for instructions on the software settings required to support [EEPROM Write protection .](https://www.raspberrypi.org/documentation/hardware/raspberrypi/booteeprom.md#%3A~%3Atext%3DEEPROM%20write%20protect)

# Chapter 3. Electrical and Mechanical

##### Mechanical

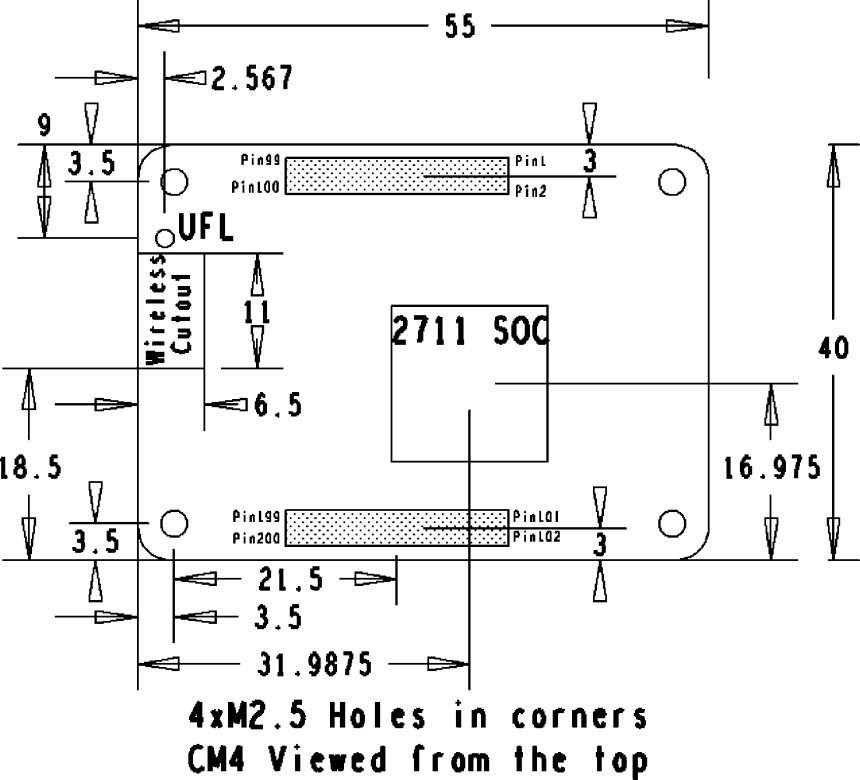
The CM4 is a compact 40 × 55mm module. The Module is 4.7mm deep, but when connected the height will be 5.078 or

6.578 mm depending on the stacking height chosen.

1. 4 × M2.5 Mounting holes (inset 3.5mm from module edge)
2. PCB thickness 1.2mm ± 10%
3. [BCM2711](https://datasheets.raspberrypi.org/bcm2711/bcm2711-peripherals.pdf) SOC height including solder balls 2.378 ± 0.11mm
4. Stacking height either:
   1. 1.5mm with mating connector (clearance under CM4 0mm) : DF40C-100DS-0.4v
   2. 3.0mm with mating connector (clearance under CM4 1.5mm): DF40HC(3.0)-100DS-0.4v

*Figuíe 4. Mechanical specification of the Raspbeííy Pi Compute Module 4*

If the on board wireless antenna is used (see [Section 2.1)](#_bookmark7) it must be orientated towards the edge of the plastic enclosure and any close by metal must have cut outs or the wireless performance will be degraded. It is suggested that there is at least 10mm clearance around the PCB antenna, but the designer must check the performance.



There must not be any metal, including ground planes, under the antenna. The ground plane cutout must be a minimum of 6.5mm × 11mm, but ideally at least 8mm × 15mm. If these requirements can’t be met wireless performance may be degraded, especially in the 2.4GHz spectrum. It is recommended that the external antenna is used where possible.

 **NOľE**

The location and arrangement of components on the Compute Module may change slightly over time due to revisions for cost and manufacturing considerations; however the maximum component heights and PCB thickness will be kept as specified.

A step file of the CM4 is available as part of the CM4 design data package, this is for guidance only and is subject to changes over time due to revisions.

##### ľheímal

The CM4 dissipates less power than the Raspberry Pi 4, Model B. The CM4 also contains less metal in the PCB and connectors and so it has less passive heat sinking than the Raspberry Pi 4, Model B. Therefore despite it consuming less power it may run warmer than the Raspberry Pi 4, Model B.

The [BCM2711](https://datasheets.raspberrypi.org/bcm2711/bcm2711-peripherals.pdf) will reduce the clock rate to try and keep its internal temperature below 85°C. So in high ambient temperatures it is possible that the clock will also be automatically throttled back. If the [BCM2711 i](https://datasheets.raspberrypi.org/bcm2711/bcm2711-peripherals.pdf)s unable to lower its internal clocks enough to bring the temperature down its case temperature will rise above 85°C. It is important that thermal solution chosen keeps the ambient temperature for the other silcon devices on the CM4 within the operating temperature range.

Operating temperature range: -20°C - +85°C Non-condensing. NB Optimal RF Wireless performance is between -20°C and

+75°C .

##### Electíical Specification

 **WARNING**

Stresses above those listed in [Table 3](#_bookmark33) may cause permanent damage to the device. This is a stress rating only; functional operation of the device under these or any other conditions above those listed in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

*ľable 3. Absolute maximum íatings*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Symbol** | **Paíameteí** | **Minimum** | **Maximum** | **Unit** |
| VIN | 5V Input Voltage | -0.5 | 6.0 | V |
| VGPIO\_ref | GPIO Voltage | -0.5 | 3.6 | V |
| Vgpio | GPIO Input voltage | -0.5 | VGPIO\_ref + 0.5v | V |

Please note that Vref is the GPIO bank voltage which must be tied to either 3.3V or 1.8v rail.

*ľable 4. DC chaíacteíistics*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Symbol** | **Paíameteí** | **Conditions** | **Minimum** | **ľypical** | **Maximum** | **Unit** |
| VIL(gpio) | Input low voltage | Vref = 3.3V | 0 | - | 0.8 | V |
| VIH(gpio) | Input high voltage | Vref = 3.3V | 2.0 | - | VGPIO\_ref | V |
| VIL(gpio) | Input low voltage | Vref = 1.8V | 0 | - | 0.35 | V |
| VIH(gpio) | Input high voltage | Vref = 1.8V | 0.65 | - | VGPIO\_ref | V |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| IIL(gpio) | Input leakage current | - | - | - | 10 | μA |
| VOL(gpio) | Output low voltage | - | - | - | 0.4 | V |
| VOH(gpio) | Output high voltage | - | VGPIO\_ref-0.4 | - | - | V |
| IO(gpio) | Output current | 1mA | 0.87 | 1.3 | - | mA |
| IO(gpio) | Output current | 2mA | 1.75 | 2.6 | - | mA |
| IO(gpio) | Output current | 3mA | 2.63 | 3.9 | - | mA |
| IO(gpio) | Output current | 4mA **Default** | 3.5 | 5.3 | - | mA |
| IO(gpio) | Output current | 5mA | 4.39 | 6.6 | - | mA |
| IO(gpio) | Output current | 6mA | 5.27 | 7.9 | - | mA |
| IO(gpio) | Output current | 7mA | 6.15 | 9.2 | - | mA |
| IO(gpio) | Output current | 8mA | 7.02 | 10.5 | - | mA |
| RPU(gpio) | Pullup resistor | Vref = 3.3V | 33 | 47 | 73 | kΩ |
| RPD(gpio) | Pulldown resistor | Vref = 3.3V | 33 | 47 | 73 | kΩ |
| RPU(gpio) | Pullup resistor | Vref = 1.8V | 18 | 47 | 73 | kΩ |
| RPD(gpio) | Pulldown resistor | Vref = 1.8V | 18 | 47 | 73 | kΩ |

*ľable 5. Poweí Consumption*

Refer to interface specifications (see [Chapter 2)](#_bookmark6) for electrical details of other interfaces.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Symbol** | **Paíameteí** | **Conditions** | **Minimum** | **ľypical** | **Maximum** | **Unit** |
| Ishutdown | Shutdown current | GLOBAL\_EN = 0v | - | 15 | - | μA |
| Ishutdown | Shutdown current | GLOBAL\_EN >2v | - | 8 | - | mA |
| Iidle | Idle current | GLOBAL\_EN >2v | - | 400 | - | mA |
| Iload | Operation current | GLOBAL\_EN >2v | - | 1400 | - | mA |

 **NOľE**

The figures in [Table 5](#_bookmark34) greatly depend on the end application.

# Chapter 4. Pin Out

*ľable 6. Pin out foí the Raspbeííy Pi Compute Module 4*

|  |  |  |
| --- | --- | --- |
| **Pin** | **Signal** | **Descíiption** |
| 1 | GND | Ground (0V) |
| 2 | GND | Ground (0V) |
| 3 | Ethernet\_Pair3\_P | Ethernet Pair 3 Positive ( connect to Transformer or MagJack) |
| 4 | Ethernet\_Pair1\_P | Ethernet Pair 1 Positive ( connect to Transformer or MagJack) |
| 5 | Ethernet\_Pair3\_N | Ethernet Pair 3 Negative ( connect to Transformer or MagJack) |
| 6 | Ethernet\_Pair1\_N | Ethernet Pair 1 Negative ( connect to Transformer or MagJack) |
| 7 | GND | Ground (0V) |
| 8 | GND | Ground (0V) |
| 9 | Ethernet\_Pair2\_N | Ethernet Pair 2 Negative ( connect to Transformer or MagJack) |
| 10 | Ethernet\_Pair0\_N | Ethernet Pair 0 Negative ( connect to Transformer or MagJack) |
| 11 | Ethernet\_Pair2\_P | Ethernet Pair 2 Positive ( connect to Transformer or MagJack) |
| 12 | Ethernet\_Pair0\_P | Ethernet Pair 0 Positive ( connect to Transformer or MagJack) |
| 13 | GND | Ground (0V) |
| 14 | GND | Ground (0V) |
| 15 | Ethernet\_nLED3 | Low Active Ethernet Activity indicator ( 3.3V signal) Typically a Green LED is connected to this pin: IOL = 8mA @ VOL< 0.4V |
| 16 | Ethernet\_SYNC\_IN | IEEE1588 SYNC Input pin ( 1.8V signal : IOL = 8mA @ VOL< 0.4V ) |
| 17 | Ethernet\_nLED2 | Low Active Ethernet speed indicator ( 3.3V signal) Typically a Yellow LED is connected to this pin. A low State indicates the 1Gbit or 100Mbit Link : IOL = 8mA @ VOL< 0.4V |
| 18 | Ethernet\_SYNC\_OUT | IEEE1588 SYNC Output pin ( 1.8V signal : IOL = 8mA @ VOL< 0.4V ) |
| 19 | Ethernet\_nLED1 | Low Active Ethernet speed indicator ( 3.3V signal) Typically a Yellow LED is connected to this pin. A low State indicates the 1Gbit or 10Mbit Link : IOL = 8mA @ VOL< 0.4V |
| 20 | EEPROM\_nWP | Leaving floating NB internally pulled up to CM4\_3.3V via 100K ( VIL <0.8V) but can be grounded to prevent writing to the on board EEPROM which stores the bootcode |
| 21 | Pi\_nLED\_Activity | Low Active Pi Activity LED. 20mA Max 5V tolerant ( VOL<0.4V). ( this is the signal that drives the Green LED on the Raspberry Pi 4, Model B ) |
| 22 | GND | Ground (0V) |
| 23 | GND | Ground (0V) |
| 24 | GPIO26 | GPIO Typically a 3.3V signal but can be a 1.8V signal by connecting GPIO\_Vref to 1.8V |
| 25 | GPIO21 | GPIO Typically a 3.3V signal but can be a 1.8V signal by connecting GPIO\_Vref to 1.8V |
| 26 | GPIO19 | GPIO Typically a 3.3V signal but can be a 1.8V signal by connecting GPIO\_Vref to 1.8V |
| 27 | GPIO20 | GPIO Typically a 3.3V signal but can be a 1.8V signal by connecting GPIO\_Vref to 1.8V |
| 28 | GPIO13 | GPIO Typically a 3.3V signal but can be a 1.8V signal by connecting GPIO\_Vref to 1.8V |
| 29 | GPIO16 | GPIO Typically a 3.3V signal but can be a 1.8V signal by connecting GPIO\_Vref to 1.8V |
| 30 | GPIO6 | GPIO Typically a 3.3V signal but can be a 1.8V signal by connecting GPIO\_Vref to 1.8V |

|  |  |  |
| --- | --- | --- |
| 31 | GPIO12 | GPIO Typically a 3.3V signal but can be a 1.8V signal by connecting GPIO\_Vref to 1.8V |
| 32 | GND | Ground (0V) |
| 33 | GND | Ground (0V) |
| 34 | GPIO5 | GPIO Typically a 3.3V signal but can be a 1.8V signal by connecting GPIO\_Vref to 1.8V |
| 35 | ID\_SC | ( BCM2711 GPIO 1) GPIO Typically a 3.3V signal but can be a 1.8V signal by connecting GPIO\_Vref to 1.8V |
| 36 | ID\_SD | ( BCM2711 GPIO 0) GPIO Typically a 3.3V signal but can be a 1.8V signal by connecting GPIO\_Vref to 1.8V |
| 37 | GPIO7 | GPIO Typically a 3.3V signal but can be a 1.8V signal by connecting GPIO\_Vref to 1.8V |
| 38 | GPIO11 | GPIO Typically a 3.3V signal but can be a 1.8V signal by connecting GPIO\_Vref to 1.8V |
| 39 | GPIO8 | GPIO Typically a 3.3V signal but can be a 1.8V signal by connecting GPIO\_Vref to 1.8V |
| 40 | GPIO9 | GPIO Typically a 3.3V signal but can be a 1.8V signal by connecting GPIO\_Vref to 1.8V |
| 41 | GPIO25 | GPIO Typically a 3.3V signal but can be a 1.8V signal by connecting GPIO\_Vref to 1.8V |
| 42 | GND | Ground (0V) |
| 43 | GND | Ground (0V) |
| 44 | GPIO10 | GPIO Typically a 3.3V signal but can be a 1.8V signal by connecting GPIO\_Vref to 1.8V |
| 45 | GPIO24 | GPIO Typically a 3.3V signal but can be a 1.8V signal by connecting GPIO\_Vref to 1.8V |
| 46 | GPIO22 | GPIO Typically a 3.3V signal but can be a 1.8V signal by connecting GPIO\_Vref to 1.8V |
| 47 | GPIO23 | GPIO Typically a 3.3V signal but can be a 1.8V signal by connecting GPIO\_Vref to 1.8V |
| 48 | GPIO27 | GPIO Typically a 3.3V signal but can be a 1.8V signal by connecting GPIO\_Vref to 1.8V |
| 49 | GPIO18 | GPIO Typically a 3.3V signal but can be a 1.8V signal by connecting GPIO\_Vref to 1.8V |
| 50 | GPIO17 | GPIO Typically a 3.3V signal but can be a 1.8V signal by connecting GPIO\_Vref to 1.8V |
| 51 | GPIO15 | GPIO Typically a 3.3V signal but can be a 1.8V signal by connecting GPIO\_Vref to 1.8V |
| 52 | GND | Ground (0V) |
| 53 | GND | Ground (0V) |
| 54 | GPIO4 | GPIO Typically a 3.3V signal but can be a 1.8V signal by connecting GPIO\_Vref to 1.8V |
| 55 | GPIO14 | GPIO Typically a 3.3V signal but can be a 1.8V signal by connecting GPIO\_Vref to 1.8V |
| 56 | GPIO3 | GPIO Typically a 3.3V signal but can be a 1.8V signal by connecting GPIO\_Vref to 1.8V. Internal 1.8K pull up to GPIO\_Vref |
| 57 | SD\_CLK | SDCARD Clock signal (only available on CM4Lite) |
| 58 | GPIO2 | GPIO Typically a 3.3V signal but can be a 1.8V signal by connecting GPIO\_Vref to 1.8V. Internal 1.8K pull up to GPIO\_Vref |
| 59 | GND | Ground (0V) |
| 60 | GND | Ground (0V) |
| 61 | SD\_DAT3 | SDCARD/eMMC Data3 signal (only available on CM4Lite) |
| 62 | SD\_CMD | SDCARD/eMMC Command signal (only available on CM4Lite) |
| 63 | SD\_DAT0 | SDCARD/eMMC Data0 signal (only available on CM4Lite) |
| 64 | SD\_DAT5 | SDCARD/eMMC Data5 signal (only available on CM4Lite) |

|  |  |  |
| --- | --- | --- |
| 65 | GND | Ground (0V) |
| 66 | GND | Ground (0V) |
| 67 | SD\_DAT1 | SDCARD/eMMC Data1 signal (only available on CM4Lite) |
| 68 | SD\_DAT4 | SDCARD/eMMC Data4 signal (only available on CM4Lite) |
| 69 | SD\_DAT2 | SDCARD/eMMC Data2 signal (only available on CM4Lite) |
| 70 | SD\_DAT7 | SDCARD/eMMC Data7 signal (only available on CM4Lite) |
| 71 | GND | Ground (0V) |
| 72 | SD\_DAT6 | SDCARD/eMMC Data6 signal (only available on CM4Lite) |
| 73 | SD\_VDD\_Override | Force SDCARD/eMMC interface to 1.8V signalling if set to 3.3V, otherwise leave unconnected. Typically only used if external eMMC is connected |
| 74 | GND | Ground (0V) |
| 75 | SD\_PWR\_ON | Output to Power switch for the SDCARD. The CM4 sets this pin High (3.3V) to signal that Power to the SDCARD should be turned on. If booting from the SDCARD is required then a pullup should also be fitted so the power defaults to on. (only available on CM4Lite) |
| 76 | Reserved | Do not Connect anything to this pin. |
| 77 | +5V (Input) | 4.75V-5.25V Main power input |
| 78 | GPIO\_VREF | Must be connected to CM4\_3.3V ( pins 84 and 86 ) for 3.3V GPIO or CM4\_1.8V ( pins 88 and 90) for 1.8V GPIO. This pin cannot be floating or connected to ground |
| 79 | +5V (Input) | 4.75V-5.25V Main power input |
| 80 | SCL0 | IIC Clock pin ( BCM2711 GPIO45) Typically used for Camera and Display Internal 1.8K pull up to CM4\_3.3V |
| 81 | +5V (Input) | 4.75V-5.25V Main power input |
| 82 | SDA0 | IIC Data pin ( BCM2711 GPIO44 ) Typically used for Camera and Display Internal 1.8K pull up to CM4\_3.3V |
| 83 | +5V (Input) | 4.75V-5.25V Main power input |
| 84 | CM4\_3.3V (Output) | 3.3V +/-2.5% Power Output max 300mA per pin for a total of 600mA. This will be powered down during power off or GLOBAL\_EN being set low |
| 85 | +5V (Input) | 4.75V-5.25V Main power input |
| 86 | CM4\_3.3V (Output) | 3.3V +/-2.5% Power Output max 300mA per pin for a total of 600mA. This will be powered down during power off or GLOBAL\_EN being set low |
| 87 | +5V (Input) | 4.75V-5.25V Main power input |
| 88 | CM4\_1.8V (Output) | 1.8V +/-2.5% Power Output max 300mA per pin for a total of 600mA. This will be powered down during power off or GLOBAL\_EN being set low |
| 89 | WL\_nDisable | Can be left floating if driven low the wireless interface will be disabled. Internal pulled up via 1.8K to CM4\_3.3V |
| 90 | CM4\_1.8V (Output) | 1.8V +/-2.5% Power Output max 300mA per pin for a total of 600mA. This will be powered down during power off or GLOBAL\_EN being set low |
| 91 | BT\_nDisable | Can be left floating if driven low the Bluetooth interface will be disabled. Internal pulled up via 1.8K to CM4\_3.3V |
| 92 | RUN\_PG | Bidirectional pin. Can be driven low ( via a 220R resistor) to Reset the CM4 CPU. As an Output a high signals Power Good and CPU running. Internally pulled up to +3.3V via 10K |

|  |  |  |
| --- | --- | --- |
| 93 | nRPIBOOT | A low on this pin forces booting from an RPI server ( e.g. PC or a Raspberry Pi) if not used leave floating. Internally pulled up via 10K to +3.3V |
| 94 | AnalogIP1 | Analogue input of the MXL7704. Typically connected to CC pin of Type C power connector |
| 95 | PI\_LED\_nPWR | Low active Output to drive Power On LED. This signal needs to be buffered. |
| 96 | AnalogIP0 | Analogue input of the MXL7704. Typically connected to CC pin of Type C power connector |
| 97 | Camera\_GPIO | Typically used to Shutdown the camera to reduce power. Reassigning this pin to another function isn’t recommended. CM4\_3.3V signalling |
| 98 | GND | Ground (0V) |
| 99 | GLOBAL\_EN | Input. Drive low to power off CM4. Internally pulled up with a 100K to +5V |
| 100 | nEXTRST | Output Driven low during reset Driven high (CM4\_3.3V) once CM4 CPU has started to boot |
| 101 | USB\_OTG\_ID | Input ( 3.3V signal ) USB OTG Pin. Internal pulled up. When grounded the CM4 becomes a USB host but the correct OS driver also needs to be used |
| 102 | PCIe\_CLK\_nREQ | Input ( 3.3V signal) PCIe Clock request pin (low to request PCI clock). Internal pulled up |
| 103 | USB\_N | USB D- |
| 104 | Reserved | Do not Connect anything to this pin. |
| 105 | USB\_P | USB D+ |
| 106 | Reserved | Do not Connect anything to this pin. |
| 107 | GND | Ground (0V) |
| 108 | GND | Ground (0V) |
| 109 | PCIe\_nRST | Output (+3.3V signal) PCIe Reset Low active |
| 110 | PCIe\_CLK\_P | PCIe Clock Out Positive (100MHz) NB AC coupling Capacitor Included on CM4 |
| 111 | VDAC\_COMP | Video DAC output (TV OUT) |
| 112 | PCIe\_CLK\_N | PCIe Clock Out Negative (100MHz) NB AC coupling Capacitor Included on CM4 |
| 113 | GND | Ground (0V) |
| 114 | GND | Ground (0V) |
| 115 | CAM1\_D0\_N | Input Camera1 D0 Negative |
| 116 | PCIe\_RX\_P | Input PCIe GEN 2 RX Positive NB External AC coupling Capacitor required |
| 117 | CAM1\_D0\_P | Input Camera1 D0 Positive |
| 118 | PCIe\_RX\_N | Input PCIe GEN 2 RX Negative NB External AC coupling Capacitor required |
| 119 | GND | Ground (0V) |
| 120 | GND | Ground (0V) |
| 121 | CAM1\_D1\_N | Input Camera1 D1 Negative |
| 122 | PCIe\_TX\_P | Output PCIe GEN 2 TX Positive NB AC coupling Capacitor Included on CM4 |
| 123 | CAM1\_D1\_P | Input Camera1 D1 Positive |
| 124 | PCIe\_TX\_N | Output PCIe GEN 2 TX Positive NB AC coupling Capacitor Included on CM4 |
| 125 | GND | Ground (0V) |
| 126 | GND | Ground (0V) |
| 127 | CAM1\_C\_N | Input Camera1 Clock Negative |

|  |  |  |
| --- | --- | --- |
| 128 | CAM0\_D0\_N | Input Camera0 D0 Negative |
| 129 | CAM1\_C\_P | Input Camera1 Clock Positive |
| 130 | CAM0\_D0\_P | Input Camera0 D0 Positive |
| 131 | GND | Ground (0V) |
| 132 | GND | Ground (0V) |
| 133 | CAM1\_D2\_N | Input Camera1 D2 Negative |
| 134 | CAM0\_D1\_N | Input Camera0 D1 Negative |
| 135 | CAM1\_D2\_P | Input Camera1 D2 Positive |
| 136 | CAM0\_D1\_P | Input Camera0 D1 Positive |
| 137 | GND | Ground (0V) |
| 138 | GND | Ground (0V) |
| 139 | CAM1\_D3\_N | Input Camera1 D3 Negative |
| 140 | CAM0\_C\_N | Input Camera0 Clock Negative |
| 141 | CAM1\_D3\_P | Input Camera1 D3 Positive |
| 142 | CAM0\_C\_P | Input Camera0 Clock Positive |
| 143 | HDMI1\_HOTPLUG | Input HDMI1 Hotplug Internally pulled down with a 100K. 5V tolerant. (It can be connected directly to a HDMI connector a small amount of ESD protection is provided on the CM4 by an on board HDMI05-CL02F3) |
| 144 | GND | Ground (0V) |
| 145 | HDMI1\_SDA | Bidir HDMI1 SDA Internally pulled up with a 1.8K. 5V tolerant. (It can be connected directly to a HDMI connector a small amount of ESD protection is provided on the CM4 by an on board HDMI05-CL02F3) |
| 146 | HDMI1\_TX2\_P | Output HDMI1 TX2 Positive |
| 147 | HDMI1\_SCL | Input HDMI1 SCL Internally pulled up with a 1.8K. 5V tolerant. (It can be connected directly to a HDMI connector a small amount of ESD protection is provided on the CM4 by an on board HDMI05-CL02F3) |
| 148 | HDMI1\_TX2\_N | Output HDMI1 TX2 Negative |
| 149 | HDMI1\_CEC | Input HDMI1 CEC Internally pulled up with a 27K. 5V tolerant. (It can be connected directly to a HDMI connector a small amount of ESD protection is provided on the CM4 by an on board HDMI05-CL02F3) |
| 150 | GND | Ground (0V) |
| 151 | HDMI0\_CEC | Input HDMI0 CEC Internally pulled up with a 27K. 5V tolerant (It can be connected directly to a HDMI connector a small amount of ESD protection is provided on the CM4 by an on board HDMI05-CL02F3) |
| 152 | HDMI1\_TX1\_P | Output HDMI1 TX1 Positive |
| 153 | HDMI0\_HOTPLUG | Input HDMI0 Hotplug Internally pulled down 100K. 5V tolerant. (It can be connected directly to a HDMI connector a small amount of ESD protection is provided on the CM4 by an on board HDMI05-CL02F3) |
| 154 | HDMI1\_TX1\_N | Output HDMI1 TX1 Negative |
| 155 | GND | Ground (0V) |
| 156 | GND | Ground (0V) |

|  |  |  |
| --- | --- | --- |
| 157 | DSI0\_D0\_N | Output Display0 D0 Negative |
| 158 | HDMI1\_TX0\_P | Output HDMI1 TX0 Positive |
| 159 | DSI0\_D0\_P | Output Display0 D0 Positive |
| 160 | HDMI1\_TX0\_N | Output HDMI1 TX0 Negative |
| 161 | GND | Ground (0V) |
| 162 | GND | Ground (0V) |
| 163 | DSI0\_D1\_N | Output Display0 D1 Negative |
| 164 | HDMI1\_CLK\_P | Output HDMI1 Clock Positive |
| 165 | DSI0\_D1\_P | Output Display0 D1 Positive |
| 166 | HDMI1\_CLK\_N | Output HDMI1 Clock Negative |
| 167 | GND | Ground (0V) |
| 168 | GND | Ground (0V) |
| 169 | DSI0\_C\_N | Output Display0 Clock Negative |
| 170 | HDMI0\_TX2\_P | Output HDMI0 TX2 Positive |
| 171 | DSI0\_C\_P | Output Display0 Clock Positive |
| 172 | HDMI0\_TX2\_N | Output HDMI0 TX2 Negative |
| 173 | GND | Ground (0V) |
| 174 | GND | Ground (0V) |
| 175 | DSI1\_D0\_N | Output Display1 D0 Negative |
| 176 | HDMI0\_TX1\_P | Output HDMI0 TX1 Positive |
| 177 | DSI1\_D0\_P | Output Display1 D0 Positive |
| 178 | HDMI0\_TX1\_N | Output HDMI0 TX1 Negative |
| 179 | GND | Ground (0V) |
| 180 | GND | Ground (0V) |
| 181 | DSI1\_D1\_N | Output Display1 D1 Negative |
| 182 | HDMI0\_TX0\_P | Output HDMI0 TX0 Positive |
| 183 | DSI1\_D1\_P | Output Display1 D1 Positive |
| 184 | HDMI0\_TX0\_N | Output HDMI0 TX0 Negative |
| 185 | GND | Ground (0V) |
| 186 | GND | Ground (0V) |
| 187 | DSI1\_C\_N | Output Display1 Clock Negative |
| 188 | HDMI0\_CLK\_P | Output HDMI0 Clock Positive |
| 189 | DSI1\_C\_P | Output Display1 Clock Positive |
| 190 | HDMI0\_CLK\_N | Output HDMI0 Clock Negative |
| 191 | GND | Ground (0V) |
| 192 | GND | Ground (0V) |
| 193 | DSI1\_D2\_N | Output Display1 D2 Negative |

|  |  |  |
| --- | --- | --- |
| 194 | DSI1\_D3\_N | Output Display1 D3 Negative |
| 195 | DSI1\_D2\_P | Output Display1 D2 Positive |
| 196 | DSI1\_D3\_P | Output Display1 D3 Positive |
| 197 | GND | Ground (0V) |
| 198 | GND | Ground (0V) |
| 199 | HDMI0\_SDA | Bidir HDMI0 SDA Internally pulled up with a 1.8K. 5V tolerant. (It can be connected directly to a HDMI connector a small amount of ESD protection is provided on the CM4 by an on board HDMI05-CL02F3) |
| 200 | HDMI0\_SCL | Bidir HDMI0 SCL Internally pulled up with a 1.8K. 5V tolerant. (It can be connected directly to a HDMI connector a small amount of ESD protection is provided on the CM4 by an on board HDMI05-CL02F3) |

All ground pins should be connected. If none of the signals on the second connector pins 101 to 200 are used then you may not fit the connector to reduce costs, but mechanical stablity needs to be considered.

The voltage on GPIO pins 0-27 must not exceed CM4\_3.3V if +3.3V signalling is used or CM4\_1.8V if +1.8V signalling is used. These pins are the same as on the 40-pin connector on the Raspberry Pi 4, Model B.

If the CM4\_1.8V rail is use to power other devices other than the GPIO\_Vref then you should ensure that in case of surprise power removal ( e.g.the +5V pin goes below +4.5V ) from the CM4, the load on the CM4\_1.8V must go to zero.

Similarly if the CM4\_3.3V rail is used to power other devices other than the GPIO\_Vref, then you should ensure that in the case surprise power removal the CM4\_3.3V rail never fails below the CM4\_1.8V rail. This is the typical case, but you should check this in your design. In the case where it does fall below the CM4\_1.8V rail, then extra circuitry is required to disconnect the CM4\_3.3V load

No reverse voltage must be applied to any pin or power up may be prevented, i.e. during power down/off no pin may have external voltage applied otherwise this may prevent power up.

##### Diffeíential Paiís

It is recommended that P/N signals within a pair are matched to better 0.15mm. Often matching between pairs is not so critical, e.g. HDMI pair to pair matching should be better than 25mm so on a typical board no extra matching is required.

* + 1. **100Ω Diffeíential paiís signal lengths**

On the CM4 all differential pairs are matched to better than 0.05mm (P/N signals).

 **NOľE**

It is recommended that pairs are also matched on the interface board.

*ľable 7. 100 Ω Diffeíential paíis signal length*

|  |  |
| --- | --- |
| **Signal** | **Length** |
| CAM0\_C\_N | 0.02 |
| CAM0\_C\_P | 0.02 |
| CAM0\_D0\_N | 0.06 |

On the CM4 pair to pairs aren’t always matched as many interfaces don’t require very accurate matching between pairs. [Table 7](#_bookmark38) documents the CM4 track length difference within each group (a non zero value is how much longer in mm that track is compared to the signal with zero length difference)

|  |  |
| --- | --- |
| CAM0\_D0\_P | 0.07 |
| CAM0\_D1\_N | 0 |
| CAM0\_D1\_P | 0.01 |
|  |  |
| CAM1\_C\_N | 0.78 |
| CAM1\_C\_P | 0.78 |
| CAM1\_D0\_N | 0.02 |
| CAM1\_D0\_P | 0.01 |
| CAM1\_D1\_N | 0.4 |
| CAM1\_D1\_P | 0.4 |
| CAM1\_D2\_N | 0.05 |
| CAM1\_D2\_P | 0.04 |
| CAM1\_D3\_N | 0.01 |
| CAM1\_D3\_P | 0 |
|  |  |
| DSI0\_C\_N | 0 |
| DSI0\_C\_P | 0 |
| DSI0\_D0\_N | 0 |
| DSI0\_D0\_P | 0 |
| DSI0\_D1\_N | 0.01 |
| DSI0\_D1\_P | 0.01 |
|  |  |
| DSI1\_C\_N | 1.28 |
| DSI1\_C\_P | 1.28 |
| DSI1\_D0\_N | 0 |
| DSI1\_D0\_P | 0.01 |
| DSI1\_D1\_N | 1.06 |
| DSI1\_D1\_P | 1.06 |
| DSI1\_D2\_N | 0.83 |
| DSI1\_D2\_P | 0.84 |
| DSI1\_D3\_N | 3.78 |
| DSI1\_D3\_P | 3.79 |
|  |  |
| HDMI0\_CLK\_N | 3.25 |
| HDMI0\_CLK\_P | 3.24 |
| HDMI0\_TX0\_N | 1.76 |
| HDMI0\_TX0\_P | 1.76 |
| HDMI0\_TX1\_N | 0.62 |

|  |  |
| --- | --- |
| HDMI0\_TX1\_P | 0.62 |
| HDMI0\_TX2\_N | 0 |
| HDMI0\_TX2\_P | 0 |
|  |  |
| HDMI1\_CLK\_N | 2.47 |
| HDMI1\_CLK\_P | 2.46 |
| HDMI1\_TX0\_N | 1.51 |
| HDMI1\_TX0\_P | 1.51 |
| HDMI1\_TX1\_N | 1 |
| HDMI1\_TX1\_P | 1 |
| HDMI1\_TX2\_N | 0 |
| HDMI1\_TX2\_P | 0.01 |
|  |  |
| Ethernet\_Pair0\_P | 5.23 |
| Ethernet\_Pair0\_N | 5.23 |
| Ethernet\_Pair1\_P | 0 |
| Ethernet\_Pair1\_N | 0 |
| Ethernet\_Pair2\_P | 3.82 |
| Ethernet\_Pair2\_N | 3.82 |
| Ethernet\_Pair3\_P | 4.29 |
| Ethernet\_Pair3\_N | 4.29 |

* + 1. **90Ω Diffeíential Paiís signal lengths**

On the CM4 all differential pairs are matched to better than 0.05mm (P/N signals).

 **NOľE**

It is recommended that pairs are also matched on the interface board.

*ľable 8. 90 Ω Diffeíential paíis signal length*

|  |  |
| --- | --- |
| **Signal** | **Length** |
| PCIe\_CLK\_P | 0.65 |
| PCIe\_CLK\_N | 0.65 |
| PCIe\_TX\_P | 0 |
| PCIe\_TX\_N | 0 |
| PCIe\_RX\_P | 0.23 |
| PCIe\_RX\_N | 0.23 |
|  |  |

However pair to pairs aren’t always matched as many interfaces don’t require very accurate matching between pairs. [Table 8](#_bookmark40) documents the CM4 track length difference within each group (a non zero value is how much longer in mm that track is compared to the signal with zero length difference)

|  |  |
| --- | --- |
| USB2\_P | 0 |
| USB2\_N | 0 |

# Chapter 5. Power

##### Poweí up sequencing

The CM4 requires a single +5V supply, and can supply up to 600mA at +3.3V and +1.8V to peripherals. All pins should not have any power applied to them before the +5V rail is applied.

If the EEPROM is to be write protected then the EEPROM\_nWP should be low before powerup.

If the CM4 is to be booted using USB then RPI\_nBOOT needs to be low within 2ms of +5V rising.

+5V should rise monotonically to 4.75V and stay above 4.75V for the entire operation of the CM4.

The power up sequence will start when both +5V rall is above 4.75V and GLOBAL\_EN rises. GLOBAL\_EN has internal RC delay so that it rises after +5V has risen. The order of events is as follows

1. +5v rises

2. GLOBAL\_EN rises

3. +3.3V rises

4. +1.8V rises at least 1mS after +3.3v

1. RUN\_PG rises at least 10mS after +1.8v
2. EXT\_nRESET rises at least 1second after RUN\_PG

##### Poweí down sequencing

The OS should be shut down to ensure that the file system remains consistent, before the power is removed. If this can’t be achieved, then a filesystem like btrfs, f2fs or overlayfs ( use raspi-config to enable it ) should be considered.

Once the OS has shutdown the +5V rail can be removed or the GLOBAL\_EN pin can be taken low to put the CM4 into the lowest power mode.

During the shutdown sequence the +1.8v will be discharged before the +3.3v rail.

##### Poweí Consumption

The exact power consumption of the CM4 will greatly depend on the tasks being run on the CM4. The lowest shutdown power consumption mode is with the GLOBAL\_EN driven low, typically is 15uA. With GLOBAL\_EN high but software shutdown the typical consumption is 8mA. Idle power cosumption is typically 400mA , but this varys considerable depending on the

Operating system. Operating power consuption is typically around 1.4A again this greatly depends on the Operating System and the Tasks being executed.

##### Regulatoí Outputs

To make it easier to interface to the CM4 the on board regulators ( +3.3v and +1.8v ) can each supply 600mA to devices connected to the CM4. The loads on these outputs isn’t taken into account in the power consumption figures.

# Appendix A: Troubleshooting

The CM4 has a number of stages of power up before the CPU starts. If there is an error at any of the stages, power up will be halted.

##### Haídwaíe Checklist

* + 1. Is the +5V supply good? Check this by pulling GLOBAL\_EN low apply and apply an external 2A load to the +5V supply. Does it stay >+4.75V including noise? Ideally it should remain >+4.9V including any noise.
    2. Remove external 2A load, but keep GLOBAL\_EN pulled low.
    3. Check the CM4 +3.3v rail is <200mV. If this is not the case there is an external power path back-feeding the CM4, either directly or indirectly. This could also occur via the digital pins, e.g Ethernet.
    4. Still with GLOBAL\_EN pulled low check the CM4 +1.8v rail is <200mW. Again if the +1.8v rail is above 200mV then there is an external path back feeding the 1.8v rail. (If nothing is connected to these pins you can ignore this check.)
    5. Remove the pull down on GLOBAL\_EN.
    6. Check GLOBAL\_EN now goes high (it internally pulled up on the CM4)
    7. Check the +3.3V supply rises to >+3.15V. If it does not, this suggests there is too much load on the +3.3V rail.
    8. Check the +1.8V rail gets to >+1.71v. If it does not, this suggests there is much load on the +1.8V rail.
    9. Check RUN\_PG goes high
    10. Check ACT\_LED starts to oscillate to indicate booting check it isn’t flashing an error code.

##### Bootloadeí

1. Connect a HDMI cable to see if the HDMI diagnostics screen appears.
2. Connect a USB serial cable to GPIO pins 14,15.
   1. See [https://www.raspberrypi.org/documentation/configuration/uart.md f](https://www.raspberrypi.org/documentation/configuration/uart.md)or details.
3. Short the nRPIBOOT pin to ground to force USB boot mode. The CM4IO board has a jumper for nRPIBOOT This can be used to enable different boot modes (e.g. network) and enable UART logging.
   1. See <https://www.raspberrypi.org/documentation/hardware/computemodule/cm-emmc-flashing.md>

##### ípi-eepíom-update

1. CM4 will not run recovery.bin from from the EMMC (or SD Card on CM4Lite). Therefore, the only way to update the bootloader EEPROM is via usbboot or self-update.

##### EEPROM Wíite píotect

The on board EEPROM can be write protected by shorting to ground EEPROM\_nWP. The CM4IO board has a jumper for

EEPROM\_nWP.

1. See <https://www.raspberrypi.org/documentation/hardware/raspberrypi/bcm2711_bootloader_config.md>

##### Ïiímwaíe

1. A 5.4 or newer kernel and the latest firmware release is required. These can be updated by using usbboot to mount the EMMC as a USB MSD device.
2. Nightly OS images are now available which contain rpi-update master firmware + kernel. Bug fixes for CM4 will normally be provided via these images except where a test/patch binary is required.

a. See <http://downloads.raspberrypi.org/nightlies/>

##### Keínel

1. The updated OS images use the new Raspberry Pi Compute Module 4 device tree file. If that is not found then the Raspberry Pi 4, Model B device tree file will be used.

a. See <https://github.com/raspberrypi/linux/blob/rpi-5.4.y/arch/arm/boot/dts/bcm2711-rpi-cm4.dts>

# Appendix B: Availability

Raspberry Pi guarantees availability of the CM4 until at least **Januaíy 2028**.

##### Suppoít

For support please see the hardware documentation section of the [Raspberry Pi website](https://www.raspberrypi.org/) and post questions to the [Raspberry Pi forum.](https://www.raspberrypi.org/forums/viewforum.php?f=98)

##### Oídeíing codes

*ľable 9. Paít Numbeí Options*

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Model** | | **Wiíeless** | | | **RAM LPDDR4** | | | **eMMC Stoíage** | |
| CM4 | | 0 = No | | | 01 = 1GByte | | | 000 = 0GByte (Lite) | |
| 1 = Yes | | | 02 = 2GByte | | | 008 = 8GByte | |
| 04 = 4GByte | | | 016 = 16GByte | |
| 08 = 8GByte | | | 032 = 32GByte | |
| **Example Paít Numbeí** | | | | | | | | | |
| CM4 | | 1 | | | 02 | | | 032 | |
| **Wiíeless** | **RAM LPDDR4** | | **Stoíage eMMC** | **RPľL** | | **Paít Numbeí** | **Oídeí Multiple** | | **RRP** |
| - | 1GB | | Lite | SC0318 | | CM4001000 | 1+ / Bulk | | $ 25.00 |
| - | 1GB | | 8GB | SC0319 | | CM4001008 | 1+ / Bulk | | $ 30.00 |
| - | 1GB | | 16GB | SC0320 | | CM4001016 | 1+ / Bulk | | $ 35.00 |
| - | 1GB | | 32GB | SC0321 | | CM4001032 | 1+ / Bulk | | $ 40.00 |
| Yes | 1GB | | Lite | SC0314 | | CM4101000 | Bulk | | $ 30.00 |
| Yes | 1GB | | 8GB | SC0315 | | CM4101008 | Bulk | | $ 35.00 |
| Yes | 1GB | | 16GB | SC0316 | | CM4101016 | Bulk | | $ 40.00 |
| Yes | 1GB | | 32GB | SC0317 | | CM4101032 | Bulk | | $ 45.00 |
| - | 2GB | | Lite | SC0287 | | CM4002000 | 1+ / Bulk | | $ 30.00 |
| - | 2GB | | 8GB | SC0288 | | CM4002008 | 1+ / Bulk | | $ 35.00 |
| - | 2GB | | 16GB | SC0289 | | CM4002016 | 1+ / Bulk | | $ 40.00 |
| - | 2GB | | 32GB | SC0290 | | CM4002032 | 1+ / Bulk | | $ 45.00 |
| Yes | 2GB | | Lite | SC0275 | | CM4102000 | 1+ / Bulk | | $ 35.00 |
| Yes | 2GB | | 8GB | SC0276 | | CM4102008 | 1+ / Bulk | | $ 40.00 |
| Yes | 2GB | | 16GB | SC0277 | | CM4102016 | 1+ / Bulk | | $ 45.00 |
| Yes | 2GB | | 32GB | SC0278 | | CM4102032 | 1+ / Bulk | | $ 50.00 |
| - | 4GB | | Lite | SC0291 | | CM4004000 | Bulk | | $ 45.00 |
| - | 4GB | | 8GB | SC0292 | | CM4004008 | Bulk | | $ 50.00 |

*ľable 10. Oídeíing Options*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| - | 4GB | 16GB | SC0293 | CM4004016 | Bulk | $ 55.00 |
| - | 4GB | 32GB | SC0294 | CM4004032 | Bulk | $ 60.00 |
| Yes | 4GB | Lite | SC0279 | CM4104000 | 1+ / Bulk | $ 50.00 |
| Yes | 4GB | 8GB | SC0280 | CM4104008 | 1+ / Bulk | $ 55.00 |
| Yes | 4GB | 16GB | SC0281 | CM4104016 | 1+ / Bulk | $ 60.00 |
| Yes | 4GB | 32GB | SC0282 | CM4104032 | 1+ / Bulk | $ 65.00 |
| - | 8GB | Lite | SC0295 | CM4008000 | Bulk | $ 70.00 |
| - | 8GB | 8GB | SC0296 | CM4008008 | Bulk | $ 75.00 |
| - | 8GB | 16GB | SC0297 | CM4008016 | Bulk | $ 80.00 |
| - | 8GB | 32GB | SC0298 | CM4008032 | Bulk | $ 85.00 |
| Yes | 8GB | Lite | SC0283 | CM4108000 | Bulk | $ 75.00 |
| Yes | 8GB | 8GB | SC0284 | CM4108008 | Bulk | $ 80.00 |
| Yes | 8GB | 16GB | SC0285 | CM4108016 | Bulk | $ 85.00 |
| Yes | 8GB | 32GB | SC0286 | CM4108032 | Bulk | $ 90.00 |

 **NOľE**

RRP was correct at time of publication and excludes taxes.

Some options have minimum ordering qualities (MOQ), please check with your supplier.

For prototyping often a higher LPDDR RAM capacity option will exist, without an MOQ. You can use the higher LPDDR RAM option, but limit it to the lower capacity by changing config.txt.

##### Packaging

Small quantities are supplied in individual cardboard boxes. These have an internal ESD coating so that a separate ESD bag isn’t required. This packaging is recyclable and reduces waste.







#Importing necessary libraries

import tensorflow as tf

from tensorflow.keras import models, layers

import matplotlib.pyplot as plt

from IPython.display import HTML

from keras import backend

#Define constants

BATCH\_SIZE = 32

IMAGE\_SIZE = 256

CHANNELS=3

EPOCHS=20

# Load dataset

import zipfile

import os

# Path to the zip file

zip\_file\_path = "/content/drive/MyDrive/RiceDiseaseDataset.zip" # zip file's location

# Directory to extract the contents

dataset\_path= "/content/RiceDiseaseDataset" # desired extraction location

# Extract the zip file

with zipfile.ZipFile(zip\_file\_path, 'r') as zip\_ref:

zip\_ref.extractall( dataset\_path)

# Check the extracted files

extracted\_files = os.listdir( dataset\_path)

print("Extracted files:", extracted\_files)

dataset= tf.keras.preprocessing.image\_dataset\_from\_directory(

dataset\_path,

shuffle=True,

image\_size=(IMAGE\_SIZE,IMAGE\_SIZE),

batch\_size=BATCH\_SIZE

)

#get class name

class\_names= dataset.class\_names

print(class\_names)

len(dataset) #determine the length of the dataset

#display information about dataset

for image\_batch, labels\_batch in dataset.take(1):

print(image\_batch.shape)

print(labels\_batch.numpy())

#display sample images from dataset

for image\_batch, label\_batch in dataset.take(1):

for i in range(12):

ax=plt.subplot(3,4,i+1)

plt.imshow(image\_batch[i].numpy().astype("uint8"))

plt.title(class\_names[label\_batch[i]])

plt.axis("off")

\*\*Partition dataset into training,validation and test sets\*\*

len(dataset)

train\_size=0.8

len(dataset)\*train\_size # Calculate the number of samples for the training set

train\_ds = dataset.take(133) # Create the training dataset by taking the first 80% of samples

len(train\_ds)

test\_ds=dataset.skip(133) # Create the test dataset by skipping the first 80% of samples

len(test\_ds)

val\_size= 0.1

len(dataset)\*val\_size # Calculate the number of samples for the validation set

val\_ds=test\_ds.take(16)

len(val\_ds)

test\_ds=test\_ds.skip(16)

len(test\_ds)

# Define a function to partition the dataset

def get\_dataset\_partitions\_tf(ds, train\_split=0.8, val\_split=0.1, test\_split=0.1, shuffle=True,

shuffle\_size=10000):

assert (train\_split + test\_split + val\_split) == 1

ds\_size = len(ds)

if shuffle:

ds = ds.shuffle(shuffle\_size, seed=12)

train\_size = int(train\_split \* ds\_size)

val\_size = int(val\_split \* ds\_size)

train\_ds = ds.take(train\_size)

val\_ds = ds.skip(train\_size).take(val\_size)

test\_ds = ds.skip(train\_size).skip(val\_size)

return train\_ds, val\_ds, test\_ds

# Partition the dataset

train\_ds, val\_ds,test\_ds= get\_dataset\_partitions\_tf(dataset)

print(len(train\_ds))

print(len(val\_ds))

print(len(test\_ds))

# Cache, shuffle, and prefetch datasets for better performance

train\_ds=train\_ds.cache().shuffle(1000).prefetch(buffer\_size=tf.data.AUTOTUNE)

val\_ds=val\_ds.cache().shuffle(1000).prefetch(buffer\_size=tf.data.AUTOTUNE)

test\_ds=test\_ds.cache().shuffle(1000).prefetch(buffer\_size=tf.data.AUTOTUNE)

#Creating a Layer for Resizing and Normalization

resize\_and\_rescale = tf.keras.Sequential([

layers.experimental.preprocessing.Resizing(IMAGE\_SIZE, IMAGE\_SIZE),

layers.experimental.preprocessing.Rescaling(1./255),

])

\*\*Data Augmentation\*\*

data\_augmentation= tf.keras.Sequential([

layers.experimental.preprocessing.RandomFlip("horizontal\_and\_vertical"),

layers.experimental.preprocessing.RandomRotation(0.2),

])

# Apply data augmentation to the training dataset

train\_ds = train\_ds.map(

lambda x, y: (data\_augmentation(x, training=True), y)

).prefetch(buffer\_size=tf.data.AUTOTUNE)

# Define input shape and number of classes

input\_shape = (BATCH\_SIZE, IMAGE\_SIZE, IMAGE\_SIZE, CHANNELS)

n\_classes = 4

# Build the CNN model

model = models.Sequential([

resize\_and\_rescale,

layers.Conv2D(32, kernel\_size = (3,3), activation='relu', input\_shape=input\_shape),

layers.MaxPooling2D((2, 2)),

layers.Conv2D(64, kernel\_size = (3,3), activation='relu'),

layers.MaxPooling2D((2, 2)),

layers.Conv2D(64, kernel\_size = (3,3), activation='relu'),

layers.MaxPooling2D((2, 2)),

layers.Conv2D(64, (3, 3), activation='relu'),

layers.MaxPooling2D((2, 2)),

layers.Conv2D(64, (3, 3), activation='relu'),

layers.MaxPooling2D((2, 2)),

layers.Conv2D(64, (3, 3), activation='relu'),

layers.MaxPooling2D((2, 2)),

layers.Flatten(),

layers.Dense(64, activation='relu'),

layers.Dense(n\_classes, activation='softmax'),

])

model.build(input\_shape=input\_shape)

model.summary()

# Compile the model

model.compile(

optimizer='adam',

loss=tf.keras.losses.SparseCategoricalCrossentropy(from\_logits=False),

metrics=['accuracy']

)

# Train the model

history = model.fit(

train\_ds,

batch\_size=BATCH\_SIZE,

validation\_data=val\_ds,

verbose=1,

epochs=EPOCHS,

)

# Evaluate the model on the test dataset

scores = model.evaluate(test\_ds)

scores

#Display model training history

history

history.params

history.history.keys()

type(history.history['loss'])

len(history.history['loss'])

history.history['loss'][:5] # show loss for first 5 epochs

#Visualize training history

acc = history.history['accuracy']

val\_acc = history.history['val\_accuracy']

loss = history.history['loss']

val\_loss = history.history['val\_loss']

plt.figure(figsize=(14, 8))

plt.subplot(1, 2, 1)

plt.plot(range(EPOCHS), acc, label='Training Accuracy')

plt.plot(range(EPOCHS), val\_acc, label='Validation Accuracy')

plt.legend(loc='lower right')

plt.title('Training and Validation Accuracy')

plt.subplot(1, 2, 2)

plt.plot(range(EPOCHS), loss, label='Training Loss')

plt.plot(range(EPOCHS), val\_loss, label='Validation Loss')

plt.legend(loc='upper right')

plt.title('Training and Validation Loss')

plt.show()

# Predict and visualize results on a sample from the test dataset

import numpy as np

for images\_batch, labels\_batch in test\_ds.take(1):

first\_image = images\_batch[0].numpy().astype('uint8')

first\_label = labels\_batch[0].numpy()

print("First image to predict")

plt.imshow(first\_image)

print("Actual label:",class\_names[first\_label])

batch\_prediction = model.predict(images\_batch)

predicted\_class = class\_names[np.argmax(batch\_prediction[0])]

print("Predicted label:", predicted\_class)

if(predicted\_class=="Brownspot"): print("Remedies:\n Chemical Pesticides : Potassium Fertilizers,

Nitrogen

Fertilizers, Magnesium Fertilizers\n Bio-pesticides : Trichoderma spp., Chaetomium spp.,

Streptomyces spp.\n Botanical Pesticides : Neem Oil, Papaya Leaf Extract, Aloe Vera Extract\n")

elif(predicted\_class=="Bacterialblight"): print("Remedies:\n Chemical Pesticides : Nitrogen

Fertilizers, Phosphorus Fertilizers\n Bio-pesticides : Bacillus subtilis, Streptomyces spp., Baculovirus\n

Botanical Pesticides : Neem Oil, Ginger Extract, Aloe Vera Extract\n")

elif(predicted\_class=="Blast"): print("Remedies:\n Chemical Pesticides : Potassium Nitrate,

Potassium Chloride, Calcium Nitrate\n Bio-pesticides : Bacillus thuringiensis (Bt), Trichoderma spp,

Pseudomonas fluorescens, Beauveria bassiana\n Botanical Pesticides : Neem Oil, Garlic Extract,

Turmeric Extract\n ")

elif(predicted\_class=="Tungro"): print("Remedies:\n Chemical Pesticides : Balanced NPK Fertilizers,

Zinc Sulfate\n Bio-pesticides : Bacillus thuringiensis (Bt), Trichoderma spp.\n Botanicals : Neem Oil,

Garlic Extract, Neem Cake (Neem Seed Kernel)\n")

# Define a function to predict labels

def predict(model, img):

img\_array = tf.keras.preprocessing.image.img\_to\_array(images[i].numpy())

img\_array = tf.expand\_dims(img\_array, 0)

predictions = model.predict(img\_array)

predicted\_class = class\_names[np.argmax(predictions[0])]

confidence = round(100 \* (np.max(predictions[0])), 2)

return predicted\_class, confidence

# Define remedies\_dict with remedies for each disease class

remedies\_dict = {

"Brownspot": [

"Chemical Pesticides: Spray Mancozeb (2.0g/lit) or Edifenphos (1ml/lit) - 2 to 3 times at 10 – 15 day intervals",

"Bio-pesticides: Seed treatment with Pseudomonas fluorescens @ 10g/kg of seed followed by

seedling dip",

"Botanical Pesticides: Neem Oil, Papaya Leaf Extract, Aloe Vera Extract"

],

"Bacterialblight": [

"Chemical Pesticides: Nitrogen Fertilizers, Phosphorus Fertilizers",

"Bio-pesticides: Bacillus subtilis, Streptomyces spp., Baculovirus",

"Botanical Pesticides: Neem Oil, Papaya Leaf Extract, Aloe Vera Extract"

],

"Blast": [

"Chemical Pesticides: Carbendazim 50WP @ 500g/ha ",

"Bio-pesticides: Dry seed treatment with Pseudomonas fluorescens talc formulation @10g/kg of seed.",

"Botanical Pesticides: Neem Oil, Garlic Extract, Turmeric Extract"

],

"Tungro": [

"Chemical Pesticides: Balanced NPK Fertilizers, Zinc Sulfate",

"Bio-pesticides: Bacillus thuringiensis (Bt), Trichoderma spp.",

"Botanicals: Neem Oil, Garlic Extract, Neem Cake (Neem Seed Kernel)"

]

}

# Redefine the predict function

def predict(model, img):

img\_array = tf.expand\_dims(img, 0) # Expand dimensions to create batch size of 1

predictions = model.predict(img\_array)

predicted\_class = class\_names[np.argmax(predictions[0])] # Get the class with highest probability

confidence = round(100 \* np.max(predictions[0]), 2) # Calculate confidence level

return predicted\_class, confidence

# Print out predicted class and confidence for each image

plt.figure(figsize=(15, 20))

for i, (images, labels) in enumerate(test\_ds.take(1)):

for j in range(4):

ax = plt.subplot(4, 3, j\*3 + 1)

plt.imshow(images[j].numpy().astype("uint8"))

# Predict class and confidence

predicted\_class, confidence = predict(model, images[j].numpy())

actual\_class = class\_names[labels[j]]

plt.title(f"Actual Label: {actual\_class}")

plt.axis("off")

ax = plt.subplot(4, 3, j\*3 + 2)

plt.text(0.4, 0.4, f"Predicted Label: {predicted\_class}\nConfidence:{confidence}%", horizontalalignment='center', verticalalignment='center', fontsize=12)

plt.axis("off")

ax = plt.subplot(4, 3, j\*3 + 3)

remedies = remedies\_dict.get(predicted\_class, [])

remedies\_text = "\n".join(remedies) if remedies else "No remedies found"

plt.text(0.5,0.5,"Remedies:\n"+remedies\_text,horizontalalignment='center',verticalalignment='center', fontsize=12, wrap=True, multialignment='center')

plt.axis("off")

# Print out predicted class and confidence

print(f"Image {j+1}: Predicted Class: {predicted\_class}, Confidence: {confidence}%")

# Adjust spacing between subplots

plt.subplots\_adjust(hspace=0.2, wspace=0.2)

plt.show()

# Save the trained model

import os

directory = '../models'

if not os.path.exists(directory):

os.makedirs(directory)

model\_version=max([int(i) for i in os.listdir("../models") + [0]])+1

model.save(f"../models/{model\_version}")

model.save("/content/sample\_data/riceleafdisease1.h5")

# Evaluate the model using confusion matrix

import numpy as np

from sklearn.metrics import confusion\_matrix

import seaborn as sns

# Predict the labels for the test dataset

test\_images = []

test\_labels = []

for images, labels in test\_ds:

test\_images.append(images.numpy())

test\_labels.append(labels.numpy())

test\_images = np.concatenate(test\_images)

test\_labels = np.concatenate(test\_labels)

predicted\_labels = np.argmax(model.predict(test\_images), axis=-1)

# Create confusion matrix

cm = confusion\_matrix(test\_labels, predicted\_labels)

# Normalize confusion matrix

cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]

# Create a figure for the plot

plt.figure(figsize=(8, 6))

# Plot the confusion matrix

sns.heatmap(cm, annot=True, fmt='.2f', cmap='Blues', xticklabels=class\_names, yticklabels=class\_names)

plt.xlabel('Predicted labels')

plt.ylabel('True labels')

plt.title('Normalized Confusion Matrix')

plt.show()

import pandas as pd

from sklearn.metrics import accuracy\_score, classification\_report

# Generate classification report

target\_names = class\_names # Use the class names as target names

report = classification\_report(test\_labels, predicted\_labels, target\_names=target\_names, output\_dict=True)

# Create a DataFrame to store the results

df = pd.DataFrame(columns=['Accuracy', 'Precision', 'Recall', 'F1-score', 'Support'])

# Extract metrics for each disease class

for class\_name, metrics in report.items():

if class\_name == 'accuracy':

df.loc['Overall Accuracy'] = [metrics, '', '', '', '']

elif class\_name in target\_names:

precision = metrics['precision']

recall = metrics['recall']

f1\_score = metrics['f1-score']

support = metrics['support']

# Check if the labels are scalar values or dictionaries

if isinstance(test\_labels[0], dict) and isinstance(predicted\_labels[0], dict):

# Extract true labels for the current class

true\_labels = [label\_dict[class\_name] for label\_dict in test\_labels]

# Extract predicted labels for the current class

pred\_labels = [label\_dict[class\_name] for label\_dict in predicted\_labels]

else:

# If labels are scalar values, use them directly

true\_labels = test\_labels

pred\_labels = predicted\_labels

# Calculate accuracy for the current class

accuracy = accuracy\_score(true\_labels, pred\_labels)

# Store metrics in the DataFrame

df.loc[class\_name] = [accuracy, precision, recall, f1\_score, support]

# Display the DataFrame

print("Classification Report:")

print(df)

**StreamLit Program Code-**

import tensorflow as tf

import streamlit as st

from PIL import Image, ImageOps

import numpy as np

from tensorflow.keras.preprocessing import image

# Load the model

loaded\_model = tf.keras.models.load\_model(r'C:\Users\PRERNA\OneDrive\Desktop\B.TechProject\riceleafdisease1.h5')

# Function to predict the label

def predict(model, img):

img\_array = tf.keras.preprocessing.image.img\_to\_array(img)

img\_array = np.expand\_dims(img\_array, axis=0)

predictions = model.predict(img\_array)

predicted\_class = class\_names[np.argmax(predictions[0])]

return predicted\_class

# Load the image

def load\_and\_preprocess\_image(image\_file):

img = Image.open(image\_file)

img = img.resize((256, 256))

return img

# Define class names

class\_names = ['Bacterialblight', 'Blast', 'Brownspot', 'Tungro']

# Define remedies for diseases

remedies = {

"Bacterialblight": """

Chemical Pesticides : Nitrogen Fertilizers, Phosphorus Fertilizers\n

Bio-pesticides : Bacillus subtilis, Streptomyces spp., Baculovirus \n

Botanical Pesticides : Neem Oil, Ginger Extract, Aloe Vera Extract\n

""",

"Blast": """

Chemical Pesticides : Carbendazim 50WP @ 500g/ha\n

Bio-pesticides : Dry seed treatment with Pseudomonas fluorescens talc formulation @10g/kg of seed.\n

Botanical Pesticides : Neem Oil, Garlic Extract, Turmeric Extract\n

""",

"Brownspot": """

Chemical Pesticides : Spray Mancozeb (2.0g/lit) or Edifenphos (1ml/lit) - 2 to 3 times at 10 - 15 day

intervals.\n

Bio-pesticides : Seed treatment with Pseudomonas fluorescens @ 10g/kg of seed followed by seedling dip\n

Botanical Pesticides : Neem Oil, Papaya Leaf Extract, Aloe Vera Extract\n

""",

"Tungro": """

Chemical Pesticides : Balanced NPK Fertilizers, Zinc Sulfate\n

Bio-pesticides : Bacillus thuringiensis (Bt), Trichoderma spp.\n

Botanicals : Neem Oil, Garlic Extract, Neem Cake (Neem Seed Kernel)\n

"""

}

# Function to predict disease

def predict\_disease(image, model):

img\_array = tf.keras.preprocessing.image.img\_to\_array(image)

img\_array = np.expand\_dims(img\_array, axis=0)

predictions = model.predict(img\_array)

predicted\_class = class\_names[np.argmax(predictions[0])]

return predicted\_class, predictions

st.write("""

# Rice Disease Detection

""")

file = st.file\_uploader("Please upload an image of a rice leaf", type=["jpg", "png"])

if file is None:

st.text("Please upload an image file")

else:

image = load\_and\_preprocess\_image(file)

st.image(image, caption='Uploaded Image', use\_column\_width=True)

predicted\_class, predictions = predict\_disease(image, loaded\_model)

confidence = np.max(predictions)

st.write("Predicted Class:", predicted\_class)

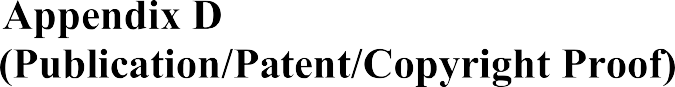
st.write("Confidence:", confidence)

st.write("Remedies:- ", remedies[predicted\_class])



|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  | 5350 |
|  |  | 1000 |
|  |  | 500 |
|  |  | 7319 |

##### Will be filing paper for Copyright

1. **Submitted the Paper**

****



##### l. Participated in Abhiyantrix 24 (16/04/2024)

**2.Attended a workshop on "Demonstration on Deep Learning Model Deployment on Raspberry Pi" (27/01/2024)**

**Rice Leaf Disease Detection and Remedies Suggestion Using Deep Learning**

# B.Tech Project ID: B-06

**Bhagyashree Marathe, Prerna Mhaisane, Atharva Tundalwar, Kalyani Patil,**

**Dr. Rajni P.K.**

**Dept. of Electronics and Telecommunication, Pimpri Chinchwad College of Engineeering, Pune.**

# INTRODUCTION

* Rice: A vital global staple powering economies, nourishing over half the world, and thriving in diverse climates with minimal water.
* Most common Rice Plant Diseases:

Brown Spot, Bacterial Leaf Blight Leaf Blast Tungro

* The detection of rice plant leaf diseases and suggesting remedies involves the utilization of a Convolutional Neural Network (CNN) approach.
* Types of Remedies for Each Disease:

Fig.3. Training and Validation graph for accuracy and loss

# METHODOLOGY

* **Data Collection:** 5932 samples illustrating four rice leaf diseases collected from Mendeley Data.
* **Data Preprocessing:** Techniques addressed missing values, removed redundancy, and downsized the dataset to 5326 samples.
* **Data Splitting:** Dataset divided into training (80%), validation (10%), and testing (10%) sets.
* **CNN Model:** Utilized to learn classification-relevant patterns from the

training dataset.

* **Disease Detection:** Relies on learned patterns to measure performance using accuracy, precision, recall, and F1 score.
* **Remedy Suggestion:** Algorithm recommends specific treatments for rice ailments to reduce yield loss and maximize crop management.

Fig.1. Block Diagram for Proposed Methodology

**RESULT**

Fig. 2. Classification Report

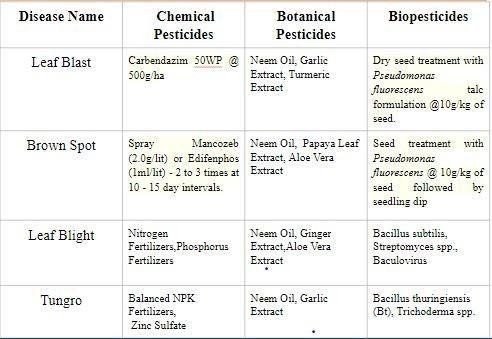
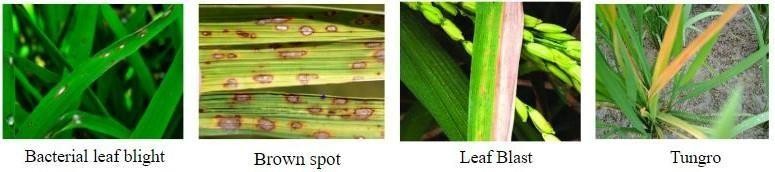
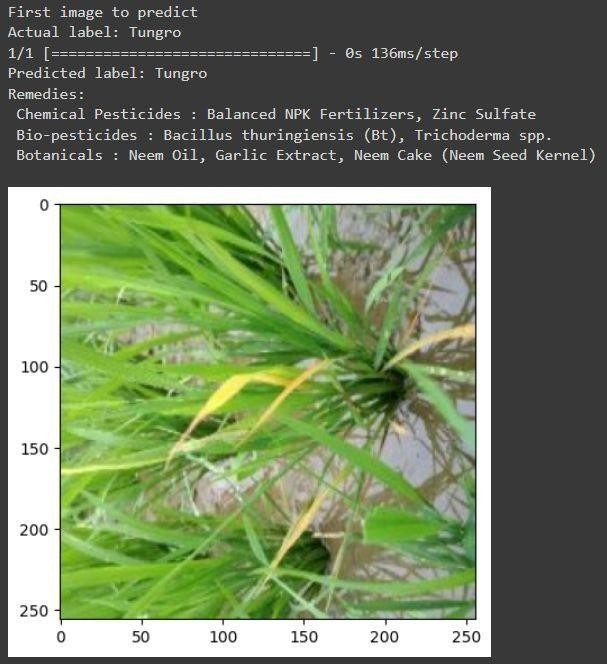
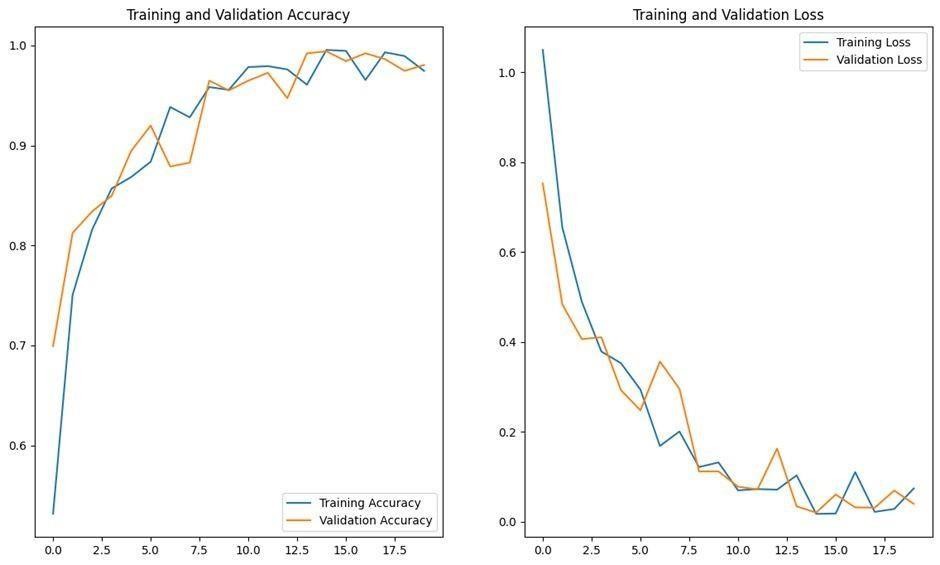
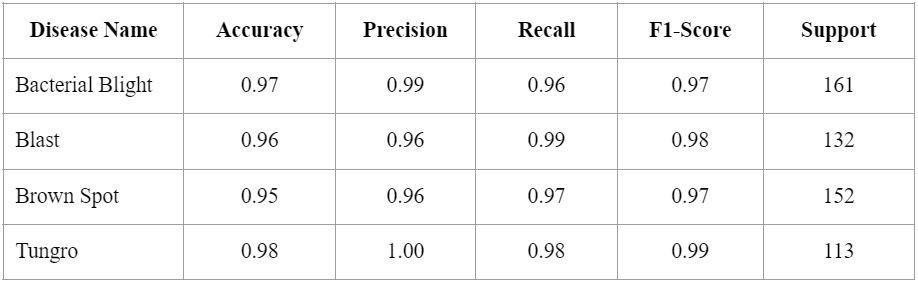
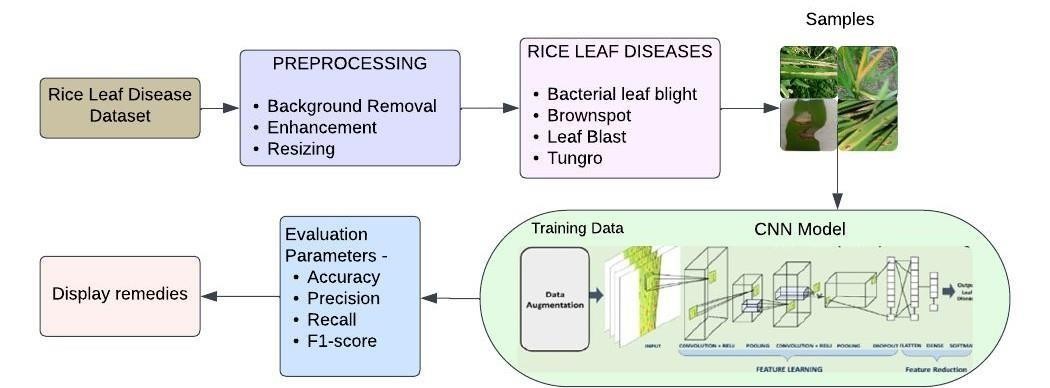
Fig. 4. Single Predicted Output Fig.5. Website Output

# CONCLUSION & FUTURE SCOPE

* Effective Disease Detection: Utilized CNNs to detect major rice diseases with a remarkable 98% overall accuracy and minimal loss.
* Rigorous Evaluation: Evaluated model performance using accuracy, precision, recall, and F1 score metrics.
* User-Friendly Web Application: Integrated detection model into a Streamlit-based web app for easy diagnosis by uploading rice leaf photos.
* Treatment Recommendations: Provided actionable treatment suggestions categorized into botanicals, conventional fertilizers, and biopesticides.
* Promotion of Sustainable Practices: Combines machine learning with intuitive web tools to enhance disease identification and promote sustainable agriculture.
* Improved Crop Resilience:: Empowers farmers to make informed decisions, manage crop health, and maximize harvests, thereby enhancing rice cultivation's resilience and productivity.

# REFERENCES

[1] Dr.Rajani P.K,Dr.Vaidehi V Deshmukh,Dr.Sheetal U. Bhandari,Dr.Roshani Raut, Dr.Reena Kharat, “Rice Leaf Disease Detection Using Convolutional Neural Network", International Journal on Recent and Innovation Trends in Computing and Communication (IJRITCC) Published by Auricle Global Society of Education and Research, Vol. 11, Issue no.10s, pp. 512–517, 7th October 2023 ISSN (Online): 2321-8169.



**RE SE ARCH P OS TE R P RE S E NTA TION TE M P LA TE © 20 19**

[**www.PosterPresentations.com**](http://www.posterpresentations.com/)