A NEW CLASSIFICATION METHOD FOR RICE VARIETY USING DEEP LEARNING

A PROJECT REPORT

Submitted by

DHANUSH KUMAR.S	**20202013**
SAM.P	**20202045**
SRIRAM.K	**20202054**

in partial fulfilment for the award of the degree

of

BACHELOR OF TECHNOLOGY

IN

INFORMATION TECHNOLOGY

PAAVAI ENGINEERING COLLEGE

(AUTONOMOUS)

ANNA UNIVERSITY : CHENNAI 600 025

MAY 2024

ANNA UNIVERSITY : CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report **"A NEW SPECIES OF RICE VARIETY CLASSIFICATION USING DEEP LEARNING"** is the bonafide work of **"DHANUSH KUMAR(20202013)"**, **"SAM.P (20202045)"** and **"SRIRAM (20202054)"** who carried out the project work under my supervision.

SIGNATURE	**SIGNATURE**
Dr.B.VENKATESAN, M.E.,Ph.D,	**MRS.VANMATHI,BE.,ME.,**
HEAD OF THE DEPARTMENT	**SUPERVISOR**
Information Technology,	Information Technology,
Paavai Engineering College,	Paavai Engineering College,
Namakkal-637018.	Namakkal-637018.

Submitted for the University Examination held on ……………………

INTERNAL EXAMINER	**EXTERNAL EXAMINER**

DECLARATION

We, **DHANUSH KUMAR (20202013), SAM (20202045) and SRIRAM (20202054),** hereby declare that the project report titled **"A NEW CLASSIFICATION METHOD FOR RICE VARIETY USING DEEP LEARNING"** done by us under the guidance of **MS.VANMATHI, BE.,ME.,** is submitted in partial fulfillment of the requirements for the award of Bachelor of Technology in Information Technology. Certified further that, to the best of our knowledge, the work reported herein does not form part of any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

1.

2.

DATE: **3.**

PLACE: PACHAL **SIGNATURE OF THE CANDIDATES**

ACKNOWLEDGEMENT

A great deal of arduous work and effort has been spent in implementing this project work. Several special people have guided us and have contributed significantly to this work and so this becomes obligatory to record our thanks to them.

We express our profound gratitude to our honourable **Chairman**, **Shri. CA. N.V.NATARAJAN, B.Com., F.C.A.,** and also to our **Correspondent**, **Smt. MANGAI NATARAJAN, M.Sc.,** for providing all necessary facilities for the successful completion of our project.

We wish to express our sincere thanks to our respected **Director-Administration, Dr. K.K.RAMASAMY, M.E., Ph.D.,** for all the blessing and help provided during the period of project work.

We would like to thank our respected **Principal, Dr. M.PREM KUMAR, M.E., Ph.D.,** for allowing us to do this project and providing required time to complete the same.

We wish to express our deep gratitude to **Dr. B.VENKATESAN, M.E., Ph.D., Head of the Department and Supervisor**, for the able guidance and useful suggestions, which helped us for completing project work in time.

We express our sincere thanks to **Professor Dr. G.MADASAMY RAJA, M.E., Ph.D., Project Coordinator** for the useful suggestions, which helped us for completing the project work in time.

We would like to extend our sincere thanks to **all our department staff members and our parents** for their advice and encouragement to do the project work with full interest and enthusiasm.

TABLE OF CONTENT

LIST OF FIGURES

ABREVATIONS

ML	**-MACHINE LEARNING**
DL	**-DEEP LEARNING**
CNN	**– CONVOLUTIONAL NEURAL NETWORKS**
RPD	**-RICE PLAQUE DETECTION**
GBA	**-GRADIENT BOOST ALGORITHM**
DLA	**-DEEP LEARNING ALGORITHMS**

DEEP LEARNING FOR RICE VARIETAL IDENTIFICATION: LEVERAGING CNNS FOR ACCURATE CLASSIFICATION

ABSTRACT

Rice varietal identification plays a crucial role in agricultural research, food safety, and quality control. In recent years, deep learning techniques, particularly Convolutional Neural Networks (CNNs), have emerged as powerful tools for image classification tasks, including the identification of different rice varieties. This paper presents a comprehensive approach to leveraging CNNs for accurate rice varietal identification. The methodology begins with data collection and preparation, involving the assembly of a diverse dataset encompassing various rice varieties under different lighting conditions and backgrounds. Supervised learning is employed, with images labelled according to their corresponding rice variety. Preprocessing techniques such as normalization and augmentation are applied to enhance dataset robustness. Next, a suitable CNN architecture is designed, drawing upon established models like sequential, or developing custom architectures tailored to the task. Emphasis is placed on maintaining spatial information and handling input images of varying sizes effectively. Techniques such as batch normalization, dropout, and appropriate activation functions are incorporated to enhance model generalization and prevent overfitting. The model is then trained on the prepared dataset, with careful consideration given to training-validation-test set splits and hyperparameter tuning. Various optimization algorithms such as stochastic gradient descent (SGD) and Adam are explored to optimize model parameters while preventing overfitting through regularization techniques.

CHAPTER 1

1. INTRODUCTION

1.1 OVERVIEW

Rice from grain products is among the products produced in many countries and consumed all over the world. Rice is priced on various parameters in the market. Texture, shape, color and fracture rate are some of these parameters. After acquiring digital images of the products, various machine learning algorithms are used to determine these parameters and perform classification operations. Machine learning algorithms ensure that large amounts of data are analyzed quickly and reliably. It is important to use such methods in rice production to improve the quality of the final product and to meet food safety criteria in an automated, economical, efficient and non-destructive way. Image processing and computer vision applications in agriculture are of interest due to their non-destructive evaluation and low cost compared to manual methods. Computer vision applications based on image processing offer advantages compared to traditional methods based on manual work. Evaluating or classifying grains by manual methods can be time-consuming and costly, as the human factor is at the forefront. In manual methods, the evaluation process may differ, as it is limited to the experience of the evaluation experts. In addition, rapid decision-making by manual methods can be difficult when an assessment is made on a large scale. Rice is the most developing crop all over India; with the increase in population, demand for rice grains has also increased. It is cultivated in almost every Asian country and exported worldwide. In India, many quality standards for rice production are made available. These include physical appearance, cooking qualities, scent, taste, smell, and efficiency difficulties. Rice has been one of the most widely consumed foods for a large part of human population. Numerous different rice varieties are imported and exported worldwide, making it the backbone of many countries’ economy. Rice seeds of different varieties can be accidentally or intentionally mixed during any of the steps in a rice production pipeline, introducing impurities. These impurities could damage the trust between rice importers and exporters, calling for the need to develop a reliable rice variety inspection system.

The use of image processing and computer vision applications in agriculture has gained interest due to their non-destructive nature and cost-effectiveness when compared to manual methods. Compared to traditional manual methods, computer vision applications based on image processing offer several advantages. Manual evaluation or classification of grains can be time-consuming and expensive, and can also be influenced by the evaluator's experience.

Furthermore, manual methods may lead to inconsistent evaluation results, especially when assessments are made on a large scale, making rapid decision-making difficult. Rice is a widely produced and consumed grain product in many countries around the world, and is priced based on various parameters in the market, such as texture, shape, color, and fracture rate. To determine these parameters and perform classification operations, digital images of rice products are acquired and various machine learning algorithms are used. These algorithms enable large amounts of data to be analyzed quickly and reliably. Using such methods in rice production is important for improving the quality of the final product and meeting food safety criteria in an automated, economical, efficient, and non-destructive manner

1.2 DEEP LEARNING

Deep learning is a subset of machine learning, which is essentially a neural network with three or more layers. These neural networks attempt to simulate the behaviour of the human brain—albeit far from matching its ability—allowing it to “learn” from large amounts of data. While a neural network with a single layer can still make approximate predictions, additional hidden layers can help to optimize and refine for accuracy. Deep learning drives many artificial intelligence (AI) applications and services that improve automation, performing analytical and physical tasks without human intervention. Deep learning technology lies behind everyday products and services (such as digital assistants, voice-enabled TV remotes, and credit card fraud detection) as well as emerging technologies (such as self-driving cars).

Deep learning is a subfield of machine learning that uses artificial neural networks to model and solve complex problems. It has emerged as one of the most promising areas of research in artificial intelligence and has been applied to a wide range of applications such as image and speech recognition, natural language processing, and robotics. Deep learning models are based on artificial neural networks that are inspired by the structure and function of the human brain. These networks consist of layers of interconnected nodes, each of which performs a mathematical operation on the input data. The output of each node is passed on to the next layer of nodes, where it is combined with the outputs of other nodes and further processed. This process continues until the output of the final layer is produced, which represents the prediction or classification of the input data.

One of the key advantages of deep learning is its ability to learn complex patterns and relationships in the data. This is achieved by using multiple layers of nodes, each of which learns a different set of features from the input data. The first layer learns low-level features such as edges and corners, while subsequent layers learn higher-level features such as textures and shapes. This hierarchical learning process enables deep learning models to capture complex patterns and relationships in the data, making them highly effective in solving complex problems. Another advantage of deep learning is its ability to learn from large amounts of data. Deep learning models require large amounts of data to train effectively, but once trained, they can make accurate predictions on new, unseen data. This makes deep learning particularly well-suited for applications such as image and speech recognition, where large amounts of labeled data are available. Deep learning has also benefited from the availability of powerful hardware such as GPUs and TPUs, which can accelerate the training and inference of deep learning models. This has enabled researchers and developers to train larger and more complex models, leading to significant improvements in performance and accuracy. Despite its many advantages, deep learning also has some limitations and challenges. One of the main challenges is the need for large amounts of labeled data. Deep learning models require labeled data to learn from, which can be difficult and expensive to obtain, especially for niche applications.

Another challenge is the interpretability of deep learning models. Deep learning models are often seen as black boxes, making it difficult to understand how they arrive at their predictions or classifications. This can be problematic in applications where interpretability is important, such as in healthcare or finance. In conclusion, deep learning has emerged as a powerful and versatile tool for solving complex problems in a wide range of applications. Its ability to learn complex patterns and relationships in the data, and its scalability to large datasets, make it particularly well-suited for applications such as image and speech recognition. However, the need for large amounts of labeled data and the interpretability of deep learning models are still challenges that need to be addressed. As deep learning continues to evolve, it is likely that these challenges will be overcome, leading to even more sophisticated and accurate models.

1.2.1 DEEP LEARNING VS MACHINE LEARNING

If deep learning is a subset of machine learning, how do they differ? Deep learning distinguishes itself from classical machine learning by the type of data that it works with and the methods in which it learns. Machine learning algorithms leverage structured, labelled data to make predictions—meaning that specific features are defined from the input data for the model and organized into tables. This doesn’t necessarily mean that it doesn’t use unstructured data; it just means that if it does, it generally goes through some pre-processing to organize it into a structured format. Deep learning eliminates some of data pre-processing that is typically involved with machine learning. These algorithms can ingest and process unstructured data, like text and images, and it automates feature extraction, removing some of the dependency on human experts. For example, let’s say that we had a set of photos of different pets, and we wanted to categorize by “cat”, “dog”, “hamster”, et cetera. Deep learning algorithms can determine which features (e.g. ears) are most important to distinguish each animal from another. In machine learning, this hierarchy of features is established manually by a human expert. Then, through the processes of gradient descent and backpropagation, the deep learning algorithm adjusts and fits itself for accuracy, allowing it to make predictions about a new photo of an animal with increased precision. Machine learning and deep learning models are capable of different types of learning as well, which are usually categorized as supervised learning, unsupervised learning, and reinforcement learning. Supervised learning utilizes labeled datasets to categorize or make predictions; this requires some kind of human intervention to label input data correctly. In contrast, unsupervised learning doesn’t require labeled datasets, and instead, it detects patterns in the data, clustering them by any distinguishing characteristics. Reinforcement learning is a process in which a model learns to become more accurate for performing an action in an environment based on feedback in order to maximize the reward.

Deep learning neural networks, or artificial neural networks, attempts to mimic the human brain through a combination of data inputs, weights, and bias. These elements work together to accurately recognize, classify, and describe objects within the data. Deep neural networks consist of multiple layers of interconnected nodes, each building upon the previous layer to refine and optimize the prediction or categorization. This progression of computations through the network is called forward propagation. The input and output layers of a deep neural network are called visible layers. The input layer is where the deep learning model ingests the data for processing, and the output layer is where the final prediction or classification is made. Another process called backpropagation uses algorithms, like gradient descent, to calculate errors in predictions and then adjusts the weights and biases of the function by moving backwards through the layers in an effort to train the model. Together, forward propagation and backpropagation allow a neural network to make predictions and correct for any errors accordingly. Over time, the algorithm becomes gradually more accurate. The above describes the simplest type of deep neural network in the simplest terms. However, deep learning algorithms are incredibly complex, and there are different types of neural networks to address specific problems or datasets. For example, Convolutional neural networks (CNNs), used primarily in computer vision and image classification applications, can detect features and patterns within an image, enabling tasks, like object detection or recognition. In 2015, a CNN bested a human in an object recognition challenge for the first time.

Recurrent neural network (RNNs) is typically used in natural language and speech recognition applications as it leverages sequential or times series data.

1.3 DEEP LEARNING APPLICATIONS

Real-world deep learning applications are a part of our daily lives, but in most cases, they are so well-integrated into products and services that users are unaware of the complex data processing that is taking place in the background. Some of these examples include the following:

Law enforcement

Deep learning algorithms can analyze and learn from transactional data to identify dangerous patterns that indicate possible fraudulent or criminal activity. Speech recognition, computer vision, and other deep learning applications can improve the efficiency and effectiveness of investigative analysis by extracting patterns and evidence from sound and video recordings, images, and documents, which helps law enforcement analyze large amounts of data more quickly and accurately.

Financial services

Financial institutions regularly use predictive analytics to drive algorithmic trading of stocks, assess business risks for loan approvals, detect fraud, and help manage credit and investment portfolios for clients.

Customer service

Many organizations incorporate deep learning technology into their customer service processes. Chatbots—used in a variety of applications, services, and customer service portals—are a straightforward form of AI. Traditional chatbots use natural language and even visual recognition, commonly found in call center-like menus. However, more sophisticated chatbot solutions attempt to determine, through learning, if there are multiple responses to ambiguous questions. Based on the responses it receives, the chatbot then tries to answer these questions directly or route the conversation to a human user. Virtual assistants like Apple's Siri, Amazon

Alexa, or Google Assistant extends the idea of a chatbot by enabling speech recognition functionality. This creates a new method to engage users in a personalized way.

Healthcare

The healthcare industry has benefited greatly from deep learning capabilities ever since the digitization of hospital records and images. Image recognition applications can support medical imaging specialists and radiologists, helping them analyze and assess more images in less time.

1.4 ADVANTAGES OF DEEP LEARNINGS

1. Feature Generation Automation

Deep learning algorithms can generate new features from among a limited number located in the training dataset without additional human intervention. This means deep learning can perform complex tasks that often require extensive feature engineering. For businesses, this means faster application or technology rollouts that deliver superior accuracy.

2. Works Well With Unstructured Data

One of the biggest draws of deep learning is its ability to work with unstructured data. In the business context, this becomes particularly relevant when you consider that the majority of business data is unstructured. Text, images, and voice are some of the most common data formats that businesses use. Classical ML algorithms are limited in their ability to analyze unstructured data, meaning this wealth of information often goes untapped. And here’s where deep learning promises to make the most impact. Training deep learning networks with unstructured data and appropriate labeling can help businesses optimize virtually every function from marketing and sales to finance.

3. Better Self-Learning Capabilities

The multiple layers in deep neural networks allow models to become more efficient at learning complex features and performing more intensive computational tasks, i.e., execute many complex operations simultaneously. It outshines machine learning in machine perception tasks (aka the ability to make sense of inputs like images, sounds, and video like a human would) that involve unstructured datasets. This is due to deep learning algorithms' ability to eventually learn from its own errors. It can verify the accuracy of its predictions/outputs and make necessary adjustments. On the other hand, classical machine learning models require varying degrees of human intervention to determine the accuracy of output.

4. Supports Parallel and Distributed Algorithms

A typical neural network or deep learning model takes days to learn the parameters that define the model. Parallel and distributed algorithms address this pain point by allowing deep learning models to be trained much faster. Models can be trained using local training (use one machine to train the model), with GPUs, or a combination of both. However, the sheer volume of the training datasets involved could mean that storing it in a single machine becomes impossible. And that's where data parallelism comes in. With data or the model itself being distributed across multiple machines, training is more effective. Parallel and distributed algorithms allow deep learning models to be trained at scale. For instance, if you were to train a model on a single computer, it could take up to 10 days to run through all the data. On the other hand, parallel algorithms can be distributed across multiple systems/computers to complete the training in less than a day. Depending on the volume of your training dataset and GPU computing power, you could use as few as two or three computers to over 20 computers to complete the training within a day.

5. Cost Effectiveness

While training deep learning models can be cost-intensive, once trained, it can help businesses cut down on unnecessary expenditure. In industries such as manufacturing, consulting, or even retail, the cost of an inaccurate prediction or product defect is massive. It often outweighs the costs of training deep learning models. Deep learning algorithms can factor in variation across learning features to reduce error margins dramatically across industries and verticals. This is particularly true when you compare the limitations of the classical machine learning model to deep learning algorithms.

6. Advanced Analytics

Deep learning, when applied to data science, can offer better and more effective processing models. Its ability to learn unsupervised drives continuous improvement in accuracy and outcomes. It also offers data scientists with more reliable and concise analysis results. The technology powers most prediction software today with applications ranging from marketing to sales, HR, finance, and more. If you use a financial forecasting tool, chances are that it uses a deep neural network. Similarly, intelligent sales and marketing automation suites also leverage deep learning algorithms to make predictions based on historical data.

7. Scalability

Deep learning is highly scalable due to its ability to process massive amounts of data and perform a lot of computations in a cost- and time-effective manner. This directly impacts productivity (faster deployment/rollouts) and modularity and portability (trained models can be used across a range of problems).

1.5 CHALLENGES IN DEEP LEARNING

While deep learning has many advantages, there are also some disadvantages to consider:

High computational cost: Training deep learning models requires significant computational resources, including powerful GPUs and large amounts of memory. This can be costly and time-consuming.

Overfitting: Overfitting occurs when a model is trained too well on the training data and performs poorly on new, unseen data. This is a common problem in deep learning, especially with large neural networks, and can be caused by a lack of data, a complex model, or a lack of regularization.

Lack of interpretability: Deep learning models, especially those with many layers, can be complex and difficult to interpret. This can make it difficult to understand how the model is making predictions and to identify any errors or biases in the model.

Dependence on data quality: Deep learning algorithms rely on the quality of the data they are trained on. If the data is noisy, incomplete, or biased, the model's performance will be negatively affected.

Data privacy and security concerns: As deep learning models often rely on large amounts of data, there are concerns about data privacy and security. Misuse of data by malicious actors can lead to serious consequences like identity theft, financial loss and invasion of privacy.

Lack of domain expertise: Deep learning requires a good understanding of the domain and the problem you are trying to solve. If the domain expertise is lacking, it can be difficult to formulate the problem and select the appropriate algorithm.

CHAPTER 2

2. LITERATURE REVIEW

2.1 TITLE: ENHANCING THE CLASSIFICATION ACCURACY OF RICE VARIETIES BY USING CONVOLUTIONAL NEURAL NETWORKS

AUTHOR: NGA TRAN-THI-KIM

Rice cultivars are planted in many countries and are the main food for a large segment of the world's population. Rice belongs to the genus Oryza. According to Vaughan, there are about 22 rice species, but only Oryza sativa and Oryza glaberrima are cultivated. To deal with the changing climate and the impacts of insects and disease, many rice varieties have been bred to deal with the challenges. Each rice variety is usually suitable for certain growing conditions like climate, soil, and water. So, selecting of a suitable variety for the growing conditions is an important part of crop. In fact, identification of rice varieties is often based on the experience of experts and farmers. However, the external appearance of rice is often similar, so it is easy to confuse one variety with another. This led to wrong selections of rice varieties for the growing conditions, so crop yields and the quality of the harvested rice will not be high. For these reasons, it is necessary to have an automatic, quick, and highly accurate classification model for classifying of rice varieties. Besides, enhancing the accuracy of classification for rice varieties helps to reduce confusion among varieties, and this leads to increases crop yields. Therefore, in this study, we classified 17 rice varieties with a similar external appearance that are popularly planted in Vietnam. In this study, three models (an ANN along with modified VGG16 and modified ResNet50 models) were applied to classify 17 rice varieties commonly planted in Vietnam. Features from images of the 17 rice varieties were extracted via the extended ILTP method to form a feature dataset. The data augmentation technique was applied to generate seven image sets with different dimensions for image dataset.

2.2 TITLE: AGE CLASSIFICATION OF RICE SEEDS IN JAPAN USING GRADIENT-BOOSTING AND ANFIS ALGORITHMS

AUTHOR: NAMAL RATHNAYAKE

Japan is a country with a population of 127 million people. For over 2000 years, rice has been a significant staple in Japan. Around 300 of the 40,000 or so distinct types of rice produced worldwide can be found in Japan. A tendency toward sustainable rice production through technical advancements has emerged, particularly in light of the rising scarcity of resources such as water and land. Researchers are driven to find new solutions to the declining or stagnant yields brought on by poor grain quality and rising production costs due to a significant reliance on agricultural inputs. At the Canadian location, seed age substantially impacted germination rate but not for seeds from the Egyptian or United Arab Emirates sites. Overall, the study emphasizes the significance of considering environmental factors when determining how seeds adapt to dormancy and germination rate. The authors looked at how alfalfa's tolerance to salt during germination rate changed with natural and artificial aging. The findings demonstrated that different aged seed lots had considerably varying salt tolerances and that seed age enhanced the amount of solute leakage that occurred after ingestion. The experiment was conducted using two steps: classification based on the rice variety and identifying the age of the seed. Since six rice varieties were available, seven classifiers were trained accordingly. Each classification task was evaluated using the confusion matrix parameters: accuracy, precision, recall, and F1-score. Moreover, the performances of the proposed algorithm were comprehensively assessed by training 13 other machine learning algorithms. The results indicate that the proposed algorithm is more capable of identifying the seed variety and age. Although other algorithms obtained better results for some occurrences, the differences in the results between other algorithms and the proposed algorithm were insignificant.

2.3 TITLE: RICE FOREIGN OBJECT CLASSIFICATION BASED ON INTEGRATED COLOR AND TEXTURAL FEATURE USING MACHINE LEARNING

AUTHOR: AJI SETIAWAN

In this study, we showed how image processing methods could be used with a camera to categorize rice that contained foreign objects using color and texture attributes. Multiclassification, SVM, decision tree, and naïve bayes method of semantic segmentation with cross-validation were used to construct foreign object rice class models. The best machine learning convolutional multivariate method to classify foreign objects from stone, gain, yellow-broken, and red-black was the SVM method, which performs better with a higher accuracy of 96.3%. For another machine learning model, the decision tree with an accuracy of 87.31% and naïve bayes 82.54%. This study aims to strengthen machine learning algorithms' classification performance to forecast four groups of foreign objects: stone, natural grain, yellow-broken and red-black. We compared the support vector machine (SVM) classifier with several other classification methods, including decision tree and naive Bayes, using color segmentation techniques, textures, and cross-validating the training data for all foreign objects. The original image was first segmented using HSV color and GLCM texture to provide more particular image features, and cross-validation was used; this method boosted SVM performance. The hybrid method attained the greatest accuracy at 96.83%. The highest Contrast value is seen in foreign grain objects, which shows a large local variance value opposite of homogeneity. Dissimilarity has similarities with contrast which is inversely proportional to homogeneity; for foreign objects, grain has the highest value with similarities such as contrast variables. Energy is highly valued when the image is not uniformly or heterogeneous in texture, and the dataset shows the highest red-black rice foreign object in this variable. The correlation indicates that the grain has the highest linear relationship value based on the grey level of the pair of pixels compared to other foreign objects.

2.4 TITLE: RICE PLAQUE DETECTION AND IDENTIFICATION BASED ON AN IMPROVED CONVOLUTIONAL NEURAL NETWORK

AUTHOR: JIAPENG CUI

Rice is one of the most important food crops in China and accounts for 30% of the country's grain output. During the growing period of rice, rice seedlings may be threatened by a variety of diseases due to the influence of many factors, such as genes and the environment. In recent years, with the influence of various factors such as climate change, rice planting systems, the selection of rice varieties and irrational drug use, the areas of rice diseases have increased significantly, accompanied by an increase in the types of diseases and aggravation of the degree of damage. The annual losses caused by rice diseases are very large; they directly threaten rice production and further threaten China's grain reserves. Therefore, disease monitoring and control of rice is a key technical approach to rice production. In the aspect of disease identification research, a new target detection model, the backbone network RlpNet, was constructed from the existing deep learning detection models. First, the model rapidly reduces the size of the feature image with two consecutive convolutions, thus improving the processing speed of the model. In the model, the overall width of the network model is improved with two tandem inception structures, which has the effect of generalizing the model and further improving the processing and running speed of the model, realizing the fast dimensionality reduction in features by continuous convolution and maximum pooling between Inception, and extracting the prominent features in the image, so as to achieve more in-depth feature extraction of multiple rice diseases, which showed that the trunk network had obvious advantages in the classification accuracy of the five rice diseases.

2.5 TITLE: LIGHTWEIGHT FEDERATED LEARNING FOR RICE LEAF DISEASE CLASSIFICATION USING NON-INDEPENDENT AND IDENTICALLY DISTRIBUTED IMAGES

AUTHOR: MEENAKSHI AGGARWAL

Rice crop fields are vulnerable to damage from diseases and pests every year, and inexperienced young farmers may struggle to identify the exact disease affecting their crops. The majority of the time, rice diseases are found using carefully supervised techniques, such as a visual inspection of the crops or lab tests. A skilled person is needed for visual inspection, which can take a lot of time. Laboratory experimentation, on the other hand, involves a lengthy process and the use of chemical reagents. In most nations, the demand for rice is anticipated to increase more quickly than the supply. Damage to the rice crop is unacceptable for this reason, regardless of the cause. It is necessary to automate the detection of rice leaf diseases to reduce crop losses. By continually scanning crops for potential infections, this automated method of disease identification will also lower labour costs. For automatically identifying rice leaf diseases, numerous researchers have put forward fascinating ideas. There are many machines learning (ML)/deep learning (DL) and internet of things (IoT) approaches available for early rice leaf diseases identification and they have proven successful in a number of different fields. However, for the past ten years, applying ML to decentralised data has been a difficult task [9]. To enable a decentralised approach, a framework that integrates multiple methods of deep learning in real-time to support the data privacy, lessen communication costs, and provide a distributed framework for training and testing is required. Lightweight federated learning is in demand right now to address important issues such as data privacy, security, access rights, and heterogeneous data access

CHAPTER 3

3. SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

The existing systems mentioned demonstrate the effectiveness of machine learning (ML) methods in agricultural applications, particularly in the categorization of wheat seeds, differentiation of crop species using canopy spectral data, and classification and detection of rice leaf diseases. The first system utilizes ML methods to categorize wheat seeds, which is crucial for various aspects of agriculture such as seed quality assessment and crop management. By employing ML techniques, this system likely analyzes features extracted from seed images to classify them into different categories based on characteristics such as size, shape, and texture. The second system employs a combination of self-attention mechanisms and 1D Convolutional Neural Networks (CNNs) to enhance precision in differentiating crop species using canopy spectral data. This approach likely involves processing spectral data collected from the canopy of crops using sensors, such as hyperspectral or multispectral imaging systems. By incorporating self-attention mechanisms, the system can effectively capture relevant spatial and spectral information from the canopy data, while 1D CNNs are utilized for feature extraction and classification. The third system focuses on the classification and detection of rice leaf diseases, a critical task in ensuring crop health and productivity. By optimizing deep neural network algorithms, this system aims to accurately classify and detect various diseases affecting rice plants based on leaf images.

3.2 DISADVANTAGES

- Classification accuracy is less
- Only support trained images
- Irrelevant features are extracted from images

3.3 PROPOSED SYSTEM

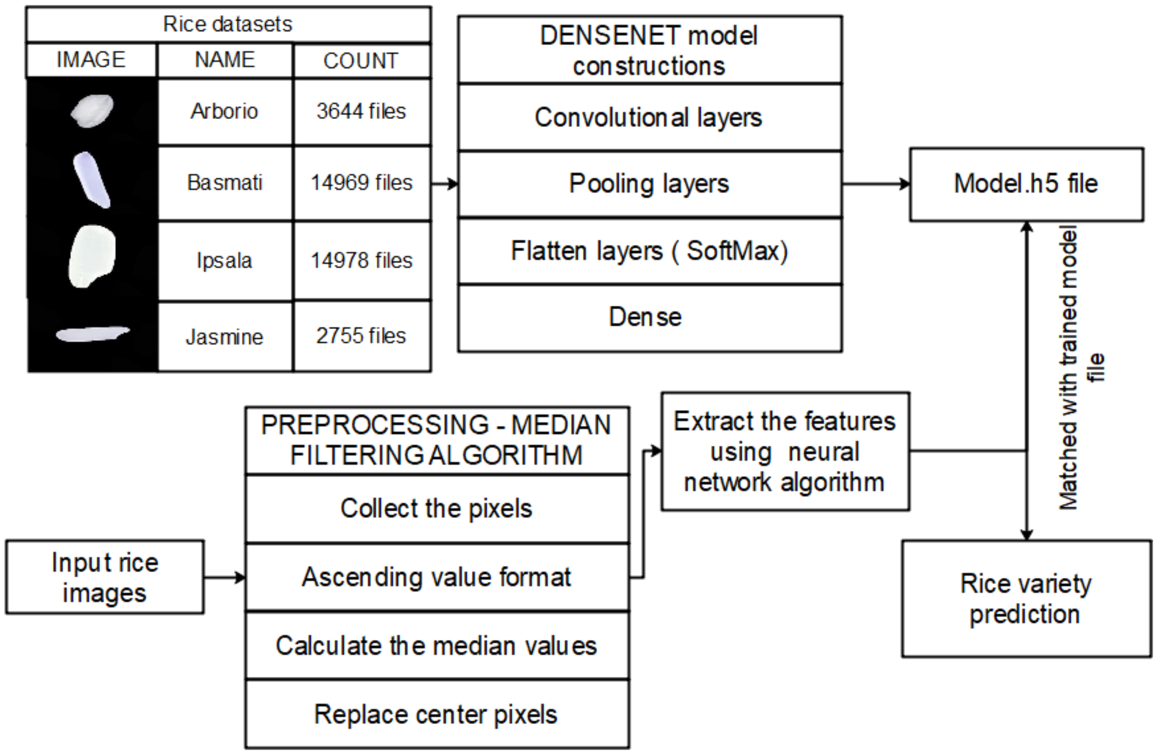
The study investigates the efficacy of DenseNet, a densely connected convolutional neural network architecture, for the task of rice variety classification based on images of rice grains. DenseNet is selected due to its efficient use of parameters and feature reuse, making it particularly advantageous for datasets with limited sample sizes, such as those common in agricultural image classification tasks. To begin the study, a comprehensive dataset comprising images of various rice varieties is constructed, ensuring diversity and representativeness. This dataset includes images captured under different lighting conditions, angles, and backgrounds to encompass the variability present in real-world scenarios. Each image is labeled according to the corresponding rice variety, providing supervised training data for the neural network. Next, DenseNet architectures are implemented and fine-tuned on the rice dataset to adapt to the intricacies of rice grain classification. The pre-trained DenseNet models, which have been previously trained on large-scale image datasets like ImageNet, serve as a starting point. Through fine-tuning, the models are adjusted to better recognize and classify rice grain images, leveraging the features learned from the general image data to improve performance on the specific task of rice variety classification. During the training process, techniques such as data augmentation, regularization, and hyperparameter optimization are employed to enhance model robustness and generalization ability. The dataset is split into training, validation, and test sets to evaluate the performance of the trained models accurately. The effectiveness of the DenseNet architectures for rice variety classification is evaluated using metrics such as accuracy, precision, recall, and F1-score. Additionally, the trained models are tested on unseen rice grain images to assess their real-world applicability and generalization capability. Overall, the study aims to demonstrate the utility of DenseNet architectures in accurately classifying rice varieties based on grain images, highlighting their potential for improving agricultural image classification tasks with limited sample sizes.

3.4 ADVANTAGES

- Improved accuracy rate
- Support multiple rice images
- Time and computational complexity is reduced

3.5 SYSTEM ARCHITECTURE

System architecture refers to the design and organization of the various components of a computer system or software application, including hardware, software, networks, and data storage. The architecture of a system can has a significant impact on its performance, scalability, security, and maintainability. Fig 3.1 defines proposed work of the system. In this architecture contains two phases such as training and testing phase. Training phase, we can collect the rice images from KAGGLE datasets. Then extract the features and build the model using DenseNet model. In testing phase, user can input the image and classify the rice types with improved efficiency.



CHAPTER 4

4. MODULE IMPLEMENTATION

4.1 MODULE LIST

- RICE IMAGE ACQUISITION
- BUILD THE MODEL
- INPUT THE IMAGE
- CLASSFICATION OF RICE

4.2 MODULES DESCRIPTION

4.2.1 RICE IMAGE ACQUISITION

Rice is one of the most important staple crops globally, providing sustenance for a significant portion of the world's population. The identification and classification of rice varieties are crucial for agricultural research, food security, and quality control. Traditional methods of varietal identification often rely on visual inspection by experts, which can be time-consuming, subjective, and prone to errors. With the advancements in computer vision and deep learning, there has been a growing interest in developing automated systems for rice variety classification based on image analysis. Set up a suitable environment for capturing images of rice grains. This setup should include appropriate lighting conditions, a stable platform for placing rice samples, and a high-resolution camera capable of capturing detailed images. Set up a suitable environment for capturing images of rice grains. This setup should include appropriate lighting conditions, a stable platform for placing rice samples, and a high-resolution camera capable of capturing detailed images. Prepare the rice samples for imaging by ensuring they are clean, free from debris, and arranged in a consistent manner. It's essential to maintain uniformity in sample presentation to minimize variability during image analysis. Arrange the rice grains in a visually appealing and informative composition for image capture. Consider capturing images from various angles and orientations to capture the full spectrum of rice grain characteristics, including size, shape, and texture.

4.2.2 BUILD THE MODEL

DenseNet, short for Densely Connected Convolutional Networks, is a deep neural network architecture specifically designed to address the vanishing gradient problem and facilitate feature reuse. The DenseNet architecture introduces dense connections between layers, where each layer receives input from all preceding layers and passes its feature maps to all subsequent layers within a dense block. This connectivity pattern enables information flow throughout the network, promoting feature reuse and facilitating gradient flow during training.

The key components of the DenseNet model architecture include:

- Dense Blocks: Dense blocks consist of multiple layers (convolutional, batch normalization, and activation) connected densely, with each layer receiving feature maps from all preceding layers within the block. This dense connectivity encourages feature reuse and facilitates the propagation of gradients during training.
- Transition Layers: Transition layers are used to reduce the dimensionality of feature maps and control the number of parameters in the network. They typically include a batch normalization layer, followed by a 1x1 convolutional layer and a downsampling operation (e.g., average pooling) to reduce the spatial dimensions of feature maps.
- Global Average Pooling: At the end of the network, global average pooling is applied to aggregate feature maps across spatial dimensions, resulting in a fixed-size feature vector that can be fed into a fully connected layer for classification.
- Dropout: Dropout regularization may be applied to the fully connected layers to prevent overfitting by randomly dropping out units during training.
- The DenseNet architecture offers several advantages:
- Parameter Efficiency: Dense connections facilitate feature reuse, allowing the network to learn more compact and efficient representations of data compared to traditional architectures.
- Gradient Flow: The dense connectivity pattern alleviates the vanishing gradient problem by providing direct paths for gradients to flow through the network during backpropagation, leading to more stable and efficient training.
- Feature Propagation: Information propagation is enhanced within dense blocks, enabling effective communication between layers and promoting the learning of highly discriminative features.
- Regularization: DenseNet naturally incorporates implicit regularization through feature reuse, reducing the need for explicit regularization techniques such as dropout.

4.2.3 INPUT THE IMAGE

To perform rice variety classification using the DenseNet model, we first preprocess the input image, typically resizing it to match the required input size of the DenseNet model and applying any necessary normalization. Next, we load a pre-trained DenseNet model, usually trained on a large dataset like ImageNet, to leverage its learned features for rice variety classification. Once the model is loaded, we feed the preprocessed image through the model to obtain predictions.

- Window Size Selection: Choose an appropriate window size (kernel size) for the median filter. The size of the window determines the extent of neighboring pixels considered for computing the median value.
- Sliding Window Operation: Slide the window over each pixel of the image, ensuring that the window covers the entire image. At each pixel position, extract the pixel values within the window.
- Median Calculation: Compute the median value of the pixel values within the window. This involves sorting the pixel values and selecting the middle value if the window size is odd, or the average of the two middle values if the window size is even.
- Replacing Pixel Values: Replace the original pixel value with the computed median value. Repeat this process for all pixels in the image.

Median filtering is a nonlinear filtering technique and is effective in preserving edges and fine details in the image, making it suitable for a wide range of image processing tasks, including image denoising, edge detection, and feature extraction. However, it may not be suitable for removing certain types of noise, such as Gaussian noise, where linear filters like Gaussian smoothing are more appropriate. Additionally, choosing an appropriate window size is crucial to balance noise reduction with preservation of image details.

4.2.4 CLASSIFICATION OF RICE

The model outputs probabilities for each rice variety class. We then interpret the model's predictions, identifying the rice variety with the highest probability as the predicted class. Finally, we display the predicted rice variety along with its associated probability score to the user. This process enables automated classification of rice varieties based on input images, leveraging the power of deep learning and pre-trained models to achieve accurate results.

CHAPTER 5

5. SYSTEM REQUIREMENTS

5.1 SOFTWARE SYSTEM CONFIGURATION

• Server Side	: Python 3.7.4(64-bit) or (32-bit)
• Client Side	: HTML, CSS, Bootstrap
• IDE	: Flask 1.1.1
• Back end	: MySQL 5.
• Server	: WampServer 2i
• OS	: Windows 10 64 –bit

5.2 HARDWARE SYSTEM CONFIGURATION

• Processor	: Intel core processor 2.6.0 GHZ
• RAM	: 4 GB
• Hard disk	: 160 GB
• Keyboard	: Standard keyboard
• Monitor	: 15-inch colour monitor

5.3. SOFTWARE DESCRIPTION

5.3.1 PYTHON

Python is a high-level, interpreted programming language that is widely used in various domains such as web development, scientific computing, data analysis, artificial intelligence, machine learning, and more. It was first released in 1991 by Guido van Rossum and has since become one of the most popular programming languages due to its simplicity, readability, and versatility. One of the key features of Python is its easy-to-learn syntax, which makes it accessible to both novice and experienced programmers. It has a large standard library that provides a wide range of modules for tasks such as file I/O, networking, regular expressions, and more. Python also has a large and active community of developers who contribute to open-source libraries and packages that extend its capabilities. Python is an interpreted language, which means that it is executed line-by-line by an interpreter rather than compiled into machine code like C or C++. This allows for rapid development and testing, as well as easier debugging and maintenance of code. Python is used for a variety of applications, including web development frameworks such as Django and Flask, scientific computing libraries such as NumPy and Pandas, and machine learning libraries such as TensorFlow and PyTorch. It is also commonly used for scripting and automation tasks due to its ease of use and readability. Overall, Python is a powerful and versatile programming language that is widely used in a variety of domains due to its simplicity, ease of use, and active community.

Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace. It provides constructs that enable clear programming on both small and large scales. In July 2018, Van Rossum stepped down as the leader in the language community. Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library. Python interpreters are available for many operating systems. CPython, the reference implementation of Python, is open-source software and has a community-based development model, as do nearly all of Python's other implementations. Python and CPython are managed by the non-profit Python Software Foundation. Rather than having all of its functionality built into its core, Python was designed to be highly extensible. This compact modularity has made it particularly popular as a means of adding programmable interfaces to existing applications. Van Rossum's vision of a small core language with a large standard library and easily extensible interpreter stemmed from his frustrations with ABC, which espoused the opposite approach. While offering choice in coding methodology, the Python philosophy rejects exuberant syntax (such as that of Perl) in favor of a simpler, less-cluttered grammar. As Alex Martelli put it: "To describe something as 'clever' is not considered a compliment in the Python culture."Python's philosophy rejects the Perl "there is more than one way to do it" approach to language design in favour of "there should be one—and preferably only one—obvious way to do it".

Python's developers strive to avoid premature optimization, and reject patches to non-critical parts of CPython that would offer marginal increases in speed at the cost of clarity. [When speed is important, a Python programmer can move time-critical functions to extension modules written in languages such as C, or use PyPy, a just-in-time compiler. CPython is also available, which translates a Python script into C and makes direct C-level API calls into the Python interpreter. An important goal of Python's developers is keeping it fun to use. This is reflected in the language's name a tribute to the British comedy group Monty Python and in occasionally playful approaches to tutorials and reference materials, such as examples that refer to spam and eggs (from a famous Monty Python sketch) instead of the standard for and bar. A common neologism in the Python community is pythonic, which can have a wide range of meanings related to program style. To say that code is pythonic is to say that it uses Python idioms well, that it is natural or shows fluency in the language, that it conforms with Python's minimalist philosophy and emphasis on readability. In contrast, code that is difficult to understand or reads like a rough transcription from another programming language is called unpythonic. Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed. Often, programmers fall in love with Python because of the increased productivity it provides. Since there is no compilation step, the edit-test-debug cycle is incredibly fast. Debugging Python programs is easy: a bug or bad input will never cause a segmentation fault. Instead, when the interpreter discovers an error, it raises an exception. When the program doesn't catch the exception, the interpreter prints a stack trace.

Python also has a large and active community of developers who contribute to a wide range of open-source libraries and tools, making it easy to find and use pre-built code to solve complex problems.

Python has a wide range of applications, including:

Data Science: Python is one of the most popular languages for data science, thanks to libraries like NumPy, Pandas, and Matplotlib that make it easy to manipulate and visualize data.

Machine Learning: Python is also widely used in machine learning and artificial intelligence, with libraries like TensorFlow, Keras, and Scikit-learn that provide powerful tools for building and training machine learning models.

Web Development: Python is commonly used in web development, with frameworks like Django and Flask that make it easy to build web applications and APIs.

Scientific Computing: Python is used extensively in scientific computing, with libraries like SciPy and SymPy that provide powerful tools for numerical analysis and symbolic mathematics.

In addition to its versatility and ease of use, Python is also known for its portability and compatibility. Python code can be run on a wide range of platforms, including Windows, macOS, and Linux, and it can be integrated with other languages like C and Java.

Overall, Python is a powerful and versatile programming language that is well-suited for a wide range of applications, from data science and machine learning to web development and scientific computing. Its simplicity, readability, and large community of developers make it an ideal choice for beginners and experts alike.

There are two attributes that make development time in Python faster than in other programming languages:

1. Python is an interpreted language, which precludes the need to compile code before executing a program because Python does the compilation in the background. Because Python is a high-level programming language, it abstracts many sophisticated details from the programming code. Python focuses so much on this abstraction that its code can be understood by most novice programmers.

2. Python code tends to be shorter than comparable codes. Although Python offers fast development times, it lags slightly in terms of execution time. Compared to fully compiling languages like C and C++, Python programs execute slower. Of course, with the processing speeds of computers these days, the speed differences are usually only observed in benchmarking tests, not in real-world operations. In most cases, Python is already included in Linux distributions and Mac OS X machines.

One of the strengths of Python is its rich ecosystem of third-party libraries and tools. These libraries provide a wide range of functionality, from scientific computing and data analysis to web development and machine learning. Some popular Python libraries and frameworks include:

NumPy: a library for numerical computing in Python, providing support for large, multi-dimensional arrays and matrices, along with a large collection of mathematical functions to operate on these arrays.

Pandas: a library for data manipulation and analysis in Python, providing support for reading and writing data in a variety of formats, as well as powerful tools for manipulating and analyzing data.

Matplotlib: a plotting library for Python that provides a variety of visualization tools, including line plots, scatter plots, bar plots, and more.

TensorFlow: an open-source machine learning library for Python that provides a variety of tools and algorithms for building and training machine learning models.

Django: a popular web framework for Python that provides a full-stack framework for building web applications, with support for everything from URL routing to user authentication and database integration.

Python's popularity has also led to a large and active community of developers who contribute to open-source projects and share code and resources online. This community provides a wealth of resources for learning Python, including tutorials, online courses, and forums for asking and answering questions.

Overall, Python is a versatile and powerful programming language that is well-suited for a wide range of applications. Its simplicity, flexibility, and wide range of libra

5.3.2 TENSORFLOW LIBARIES IN PYTHON

TensorFlow is an open-source machine learning framework developed by Google Brain Team. It is one of the most popular libraries for building and training machine learning models, especially deep neural networks. TensorFlow allows developers to build complex models with ease, including image and speech recognition, natural language processing, and more. One of the key features of TensorFlow is its ability to handle large-scale datasets and complex computations, making it suitable for training deep neural networks. It allows for parallelization of computations across multiple CPUs or GPUs, allowing for faster training times. TensorFlow also provides a high-level API called Keras that simplifies the process of building and training models. TensorFlow offers a wide range of tools and libraries that make it easy to integrate with other Python libraries and frameworks. It has built-in support for data preprocessing and visualization, making it easy to prepare data for training and analyze model performance. One of the major advantages of TensorFlow is its ability to deploy models to a variety of platforms, including mobile devices and the web.

Graph-based computation: TensorFlow uses a graph-based computation model, which allows for efficient execution of computations across multiple devices and CPUs/GPUs.

Automatic differentiation: TensorFlow provides automatic differentiation, which allows for efficient computation of gradients for use in backpropagation algorithms.

High-level APIs: TensorFlow provides high-level APIs, such as Keras, that allow developers to quickly build and train complex models with minimal code.

Preprocessing and data augmentation: TensorFlow provides a range of tools for preprocessing and data augmentation, including image and text preprocessing, data normalization, and more.

Distributed training: TensorFlow supports distributed training across multiple devices, CPUs, and GPUs, allowing for faster training times and more efficient use of resources.

Model deployment: TensorFlow allows for easy deployment of models to a variety of platforms, including mobile devices and the web.

Visualization tools: TensorFlow provides a range of visualization tools for analyzing model performance, including TensorBoard, which allows for real-time visualization of model training and performance.

5.3.3 PYCHARM

PyCharm is an integrated development environment (IDE) for Python programming language, developed by JetBrains. PyCharm provides features such as code completion, debugging, code analysis, refactoring, version control integration, and more to help developers write, test, and debug their Python code efficiently. PyCharm is available in two editions: Community Edition (CE) and Professional Edition (PE). The Community Edition is a free, open-source version of the IDE that provides basic functionality for Python development. The Professional Edition is a paid version of the IDE that provides advanced features such as remote development, web development, scientific tools, database tools, and more. PyCharm is available for Windows, macOS, and Linux operating systems. It supports Python versions 2.7, 3.4, 3.5, 3.6, 3.7, 3.8, 3.9, and 3.10.

Features:

- Intelligent code completion
- Syntax highlighting
- Code inspection
- Code navigation and search
- Debugging
- Testing
- Version control integration
- Web development support
- Scientific tools support
- Database tools support

Integration with other JetBrains tools

PyCharm's code completion feature can help speed up development by automatically suggesting code based on context and previously written code. It also includes a debugger that allows developers to step through code, set breakpoints, and inspect variables. PyCharm has integration with version control systems like Git, Mercurial, and Subversion. It also supports virtual environments, which allow developers to manage different Python installations and packages in isolated environments. The IDE also has features specifically geared towards web development, such as support for popular web frameworks like Django, Flask, and Pyramid. It includes tools for debugging, testing, and profiling web applications. PyCharm also provides scientific tools for data analysis, visualization, and scientific computing, such as support for NumPy, SciPy, and matplotlib. It also includes tools for working with databases, such as PostgreSQL, MySQL, and Oracle. Overall, PyCharm is a powerful and feature-rich IDE that can greatly increase productivity for Python developers.

Customization:

PyCharm allows developers to customize the IDE to their liking. Users can change the color scheme, fonts, and other settings to make the IDE more comfortable to use. PyCharm also supports plugins, which allow developers to extend the IDE with additional features.

Collaboration:

PyCharm makes it easy for developers to collaborate on projects. It supports integration with popular collaboration tools such as GitHub, Bitbucket, and GitLab. It also includes features for code reviews, task management, and team communication.

Education:

PyCharm provides a learning environment for Python programming language. PyCharm Edu is a free, open-source edition of PyCharm that includes interactive courses and tutorials for learning Python. It provides an easy-to-use interface for beginners and includes features such as code highlighting, autocompletion, and error highlighting.

Support:

PyCharm has an active community of users who provide support through forums and social media. JetBrains also provides comprehensive documentation, tutorials, and training courses for PyCharm. For users who need more personalized support, JetBrains offers a paid support plan that includes email and phone support.

Pricing:

PyCharm Community Edition is free and open-source. PyCharm Professional Edition requires a paid license, but offers a 30-day free trial. JetBrains also offers a subscription-based pricing model that includes access to all JetBrains IDEs and tools.

Integrations:

PyCharm integrates with a wide range of tools and technologies commonly used in Python development. It supports popular Python web frameworks like Flask, Django, Pyramid, and web2py. It also integrates with tools for scientific computing like NumPy, SciPy, and pandas. PyCharm also supports popular front-end technologies such as HTML, CSS, and JavaScript.

Performance:

PyCharm is known for its fast and reliable performance. It uses a combination of static analysis, incremental compilation, and intelligent caching to provide fast code completion and navigation. PyCharm also has a memory profiler that helps identify and optimize memory usage in Python applications.

Ease of Use:

PyCharm provides an intuitive and easy-to-use interface for developers. It has a well-organized menu structure, clear icons, and easy-to-navigate tabs. PyCharm also provides a variety of keyboard shortcuts and customizable keymaps that allow users to work efficiently without constantly switching between the mouse and keyboard.

Community:

PyCharm has a large and active community of developers who contribute to the development of the IDE. The PyCharm Community Edition is open-source, which means that anyone can contribute to its development. The PyCharm user community is also active in providing support, tips, and tutorials through forums, blogs, and social media.

5.4 MY SQL

MySQL is the world's most used open source relational database management system (RDBMS) as of 2008 that run as a server providing multi-user access to a number of databases. The MySQL development project has made its source code available under the terms of the GNU General Public License, as well as under a variety of proprietary agreements. MySQL was owned and sponsored by a single for-profit firm, the Swedish company MySQL AB, now owned by Oracle Corporation.

MySQL is a popular choice of database for use in web applications, and is a central component of the widely used LAMP open source web application software stack—LAMP is an acronym for "Linux, Apache, MySQL, Perl/PHP/Python." Free-software-open-source projects that require a full-featured database management system often use MySQL.For commercial use, several paid editions are available, and offer additional functionality. Applications which use MySQL databases include: TYPO3, Joomla, Word Press, phpBB, MyBB, Drupal and other software built on the LAMP software stack. MySQL is also used in many high-profile, large-scale World Wide Web products, including Wikipedia, Google(though not for searches), Imagebook Twitter, Flickr, Nokia.com, and YouTube.

Inter images

MySQL is primarily an RDBMS and ships with no GUI tools to administer MySQL databases or manage data contained within the databases. Users may use the included command line tools, or use MySQL "front-ends", desktop software and web applications that create and manage MySQL databases, build database structures, back up data, inspect status, and work with data records. The official set of MySQL front-end tools, MySQL Workbench is actively developed by Oracle, and is freely available for use.

Graphical

The official MySQL Workbench is a free integrated environment developed by MySQL AB that enables users to graphically administer MySQL databases and visually design database structures. MySQL Workbench replaces the previous package of software, MySQL GUI Tools. Similar to other third-party packages, but still considered the authoritative MySQL frontend, MySQL Workbench lets users manage database design & modeling, SQL development (replacing MySQL Query Browser) and Database administration (replacing MySQL Administrator).MySQL Workbench is available in two editions, the regular free and open source Community Edition which may be downloaded from the MySQL website, and the proprietary Standard Edition which extends and improves the feature set of the Community Edition.

CHAPTER 6

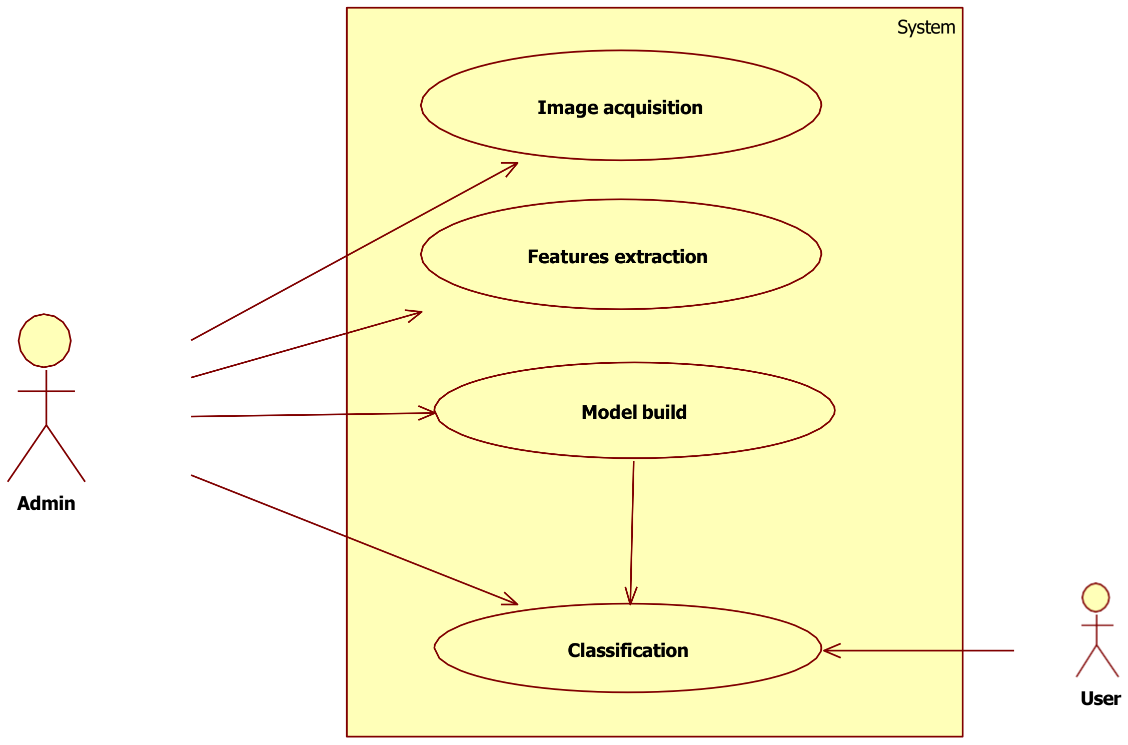
6. SYSTEM DESIGN

6.1 UML DIAGRAMS

Unified Modeling Language (UML) diagrams serve as powerful tools for visualizing and designing complex systems, offering insights into their structure, behavior, and interactions. In the context of a rice classification system, various UML diagrams can be employed to elucidate different aspects of the system.

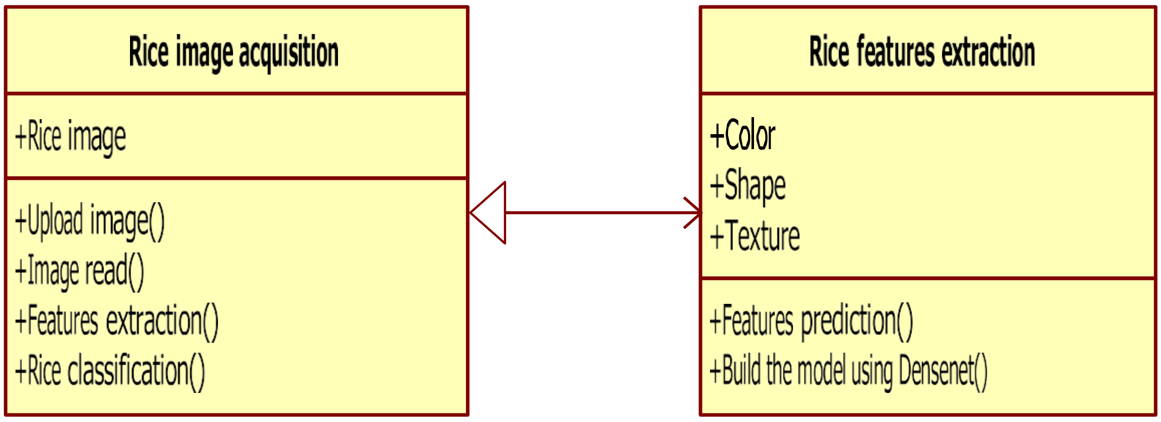
6.2 USECASE DIAGRAM

Use case diagrams are usually referred to as behavior diagrams used to describe a set of actions (use cases) that some system or systems (subject) should or can perform in collaboration with one or more external users of the system (actors). Each use case should provide some observable and valuable result to the actors or other stakeholders of the system.



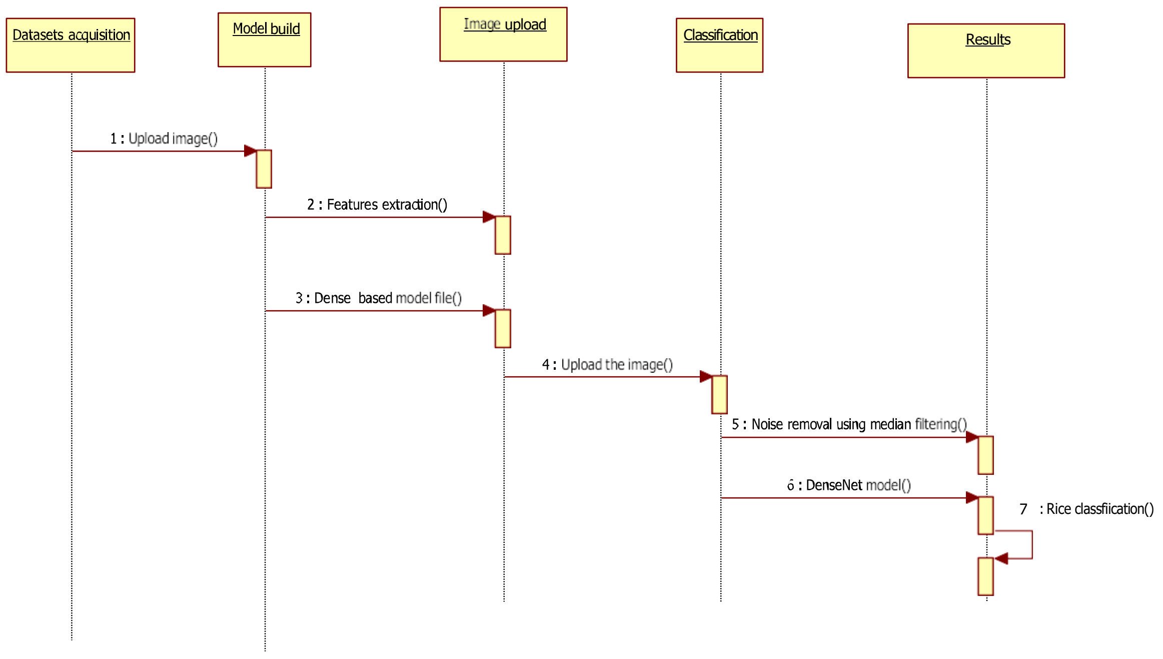
6.3 CLASS DIAGRAM

The class diagram is the main building block of object-oriented modeling. It is used both for general conceptual modeling of the systematic of the application, and for detailed modeling translating the models into programming code. Class diagrams can also be used for data modeling. The classes in a class diagram represent both the main elements, interactions in the application, and the classes to be programmed.



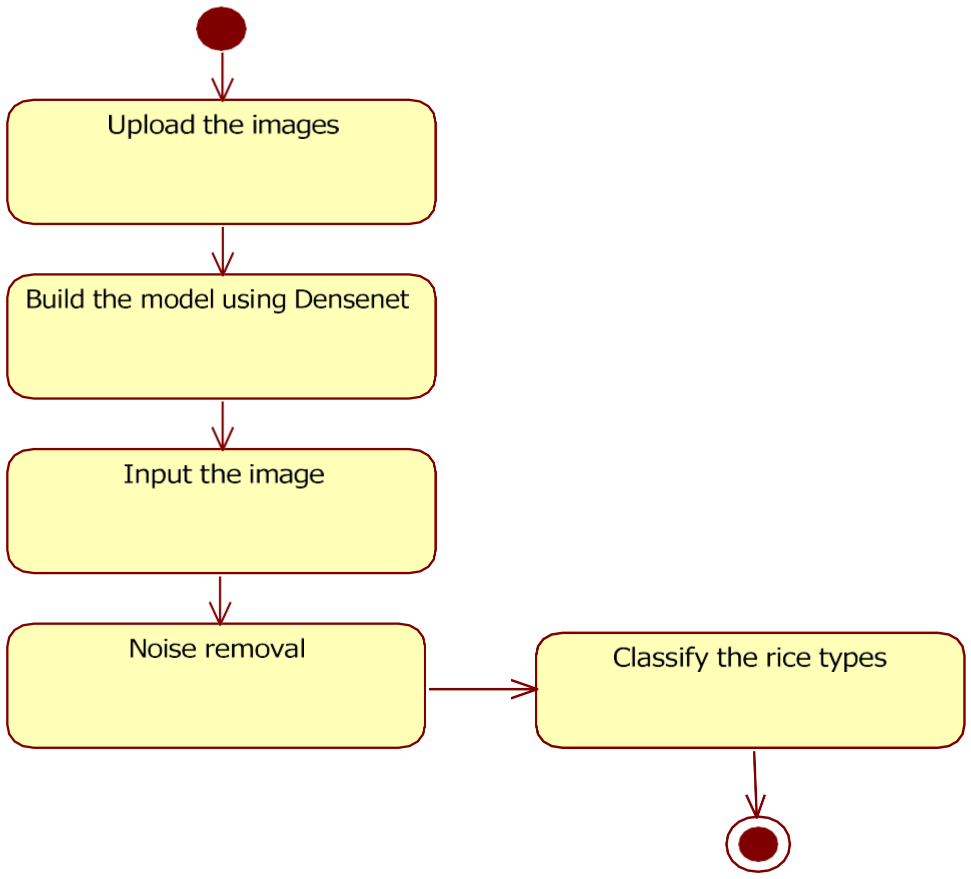
6.4 SEQUENCE DIAGRAM

A Sequence diagram is an interaction diagram that shows how objects operate with one another and in what order. It is a construct of a message sequence chart. A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the Logical View of the system under development. Sequence diagrams are sometimes called event diagrams or event scenarios.



6.5 DEPLOYMENT DIAGRAM

A deployment diagram shows components and artifacts in relation to where they are used in the deployed system. A component diagram defines the composition of components and artifacts in the system. A deployment diagram is just a special kind of class diagram, which focuses on a system's nodes. Graphically, a deployment diagram is a collection of vertices and arcs.



CHAPTER 7
7. SYSTEM IMPLEMENTATION

7.1 INPUT DESIGN

Input design is a critical aspect of designing any system, including a rice variety classification system. In the context of such a system, input design involves determining how users will input rice images for classification and ensuring that the process is intuitive, efficient, and user-friendly. Here's an explanation of input design considerations for a rice classification system:

Image Upload Interface: The primary input mechanism for the system is likely to be an image upload interface, where users, such as dermatologists or patients, can submit rice images for classification. This interface should be straightforward and easy to use, allowing users to upload images from their devices or capture images using a webcam or mobile device.

File Format and Size Restrictions: Define acceptable file formats (e.g., JPEG, PNG) and size restrictions for uploaded images to ensure compatibility with the classification model and efficient processing. Provide clear instructions to users regarding acceptable formats and sizes to avoid errors during image submission.

Error Handling and Validation: Implement error handling and validation mechanisms to verify the correctness and integrity of uploaded images. This may include checks for file format, size, and resolution, as well as validation of metadata such as patient information or image capture date.

7.2 OBJECTIVES

The objectives of a rice classification system are multifaceted, aiming to address various needs and goals within the context of dermatological diagnosis and patient care. Here are some key objectives:

Automated Diagnosis: Develop a system capable of automatically classifying rice images into different disease categories, reducing reliance on manual inspection by dermatologists and streamlining the diagnostic process.

Accuracy and Reliability: Ensure that the classification system achieves high accuracy and reliability in identifying different rice types, minimizing misdiagnosis and improving patient outcomes.

User-Friendly Interface: Create a user-friendly interface for uploading rice images and accessing classification results, catering to the needs of both dermatologists and patients. The interface should be intuitive, easy to navigate, and require minimal training to use effectively.

7.3 OUTPUT DESIGN

Output design is crucial for a rice classification system as it determines how the classification results are presented to users, such as dermatologists or patients. Here are some key considerations for designing the output of the system:

- Input Image Display: Display the input rice image that is being classified. This allows users to see the image being analysed and provides context for the classification results.
- Predicted Rice Variety: Display the predicted rice variety along with the associated probability score. This information indicates the rice variety that the classification model has identified as most likely based on the input image.
- Probability Distribution: Optionally, visualize the probability distribution of the predicted rice varieties. This can be presented as a bar chart or a pie chart, showing the probabilities assigned to each rice variety by the classification model.
- Confidence Level: Provide an indication of the confidence level or certainty associated with the prediction. This could be displayed as a percentage or a qualitative assessment (e.g., high confidence, medium confidence, low confidence).
- Additional Information: Optionally, display additional information about the classification process, such as the model used, any preprocessing steps applied to the image, and the number of rice varieties considered in the classification task.
- User Interaction: Allow users to interact with the classification results, such as zooming in on the input image, viewing detailed information about the predicted rice variety, or exploring alternative classification results if available.

CHAPTER 8

8. FEASIBILITY STUDY

8.1 OBJECTIVES

The purpose of this chapter is to introduce the reader to feasibility studies, project appraisal, and investment analysis. Feasibility studies are an example of systems analysis. A system is a description of the relationships between the inputs of labour, machinery, materials and management procedures, both within an organisation and between an organisation and the outside world.

During the planning and execution stages of an audit, it’s important to have a clear understanding of what the objectives of the audit include. Companies should strive to align their business objectives with the objectives of the audit. This will ensure that time and resources spent will help achieve a strong internal control environment and lower the risk of a qualified opinion.

Objectives of Feasibility Study

- To explain present situation of the automation.
- To find out if a system development project can be done is possible.
- To find out whether the final product will benefit end user.
- To suggest the possible alternative solutions.

8.2 TECHNICAL FEASIBILITY

Technical Feasibility assessment focuses on the technical resources available to the organization. It helps organizations determine whether the technical resources meet capacity and whether the technical team is capable of converting the ideas into working systems. In technical feasibility the following issues are taken into consideration.

- Whether the required technology is available or not
- Whether the required resources are available - Manpower- programmers, testers & debuggers, Software and hardware

Once the technical feasibility is established, it is important to consider the monetary factors also. Since it might happen that developing a particular system may be technically possible but it may require huge investments and benefits may be less. For evaluating this, economic feasibility of the proposed system is carried out.

8.3 OPERATIONAL FEASIBILITY

Operational Feasibility is depended on human resources available for the project and involves projecting whether the system will be used if it is developed and implemented. Operational feasibility is a measure of how well a proposed system solves the problems, and takes advantage of the opportunities identified during scope definition and how it satisfies the requirements analysis phase of system development. Operational feasibility reviews the willingness of the organization to support the proposed system. This is probably the most difficult of the feasibilities to gauge. In order to determine this feasibility, it is important to understand the management commitment to the proposed project. If the request was initiated by management, it is likely that there is management support and the system will be accepted and used. However, it is also important that the employee base will be accepting of the change.

8.4 ECONOMICAL FEASIBILITY

Economic feasibility analysis is the most commonly used method for determining the efficiency of a new project. It is also known as cost analysis. It helps in identifying profit against investment expected from a project. Cost and time are the most essential factors involved in this field of study. For any system if the expected benefits equal or exceed the expected costs, the system can be judged to be economically feasible. In economic feasibility, cost benefit analysis is done in which expected costs and benefits are evaluated. Economic analysis is used for evaluating the effectiveness of the proposed system.

CHAPTER 9

9. TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of tests. Each test type addresses a specific testing requirement.

9.1 SOFTWARE TESTING STRATEGIES

Testing involves

- Unit Testing
- Functional Testing
- Acceptance Testing
- Integration Testing

9.1.1 UNIT TESTING

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application. It done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

9.1.2 FUNCTIONAL TESTING

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input	:	identified classes of valid input must be accepted.
Invalid Input	:	identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

9.1.3 ACCEPTANCE TESTING

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

9.1.4 INTEGRATION TESTING

Integration testing is a software testing technique that involves testing the interaction between different software components or modules to ensure that they work together correctly. The main goal of integration testing is to verify that the individual components of the software system can work together as expected and that the system as a whole meets the functional and non-functional requirements.

CHAPTER 10

10. APPENDIX

10.1 SOURCE CODE

MODEL.PY

```
import matplotlib.pyplot as plt

import warnings

import seaborn as sns

import numpy

warnings.filterwarnings('ignore')

batch_size = 32




from tensorflow.keras.preprocessing.image import ImageDataGenerator


train_datagen = ImageDataGenerator(rescale=1/255)



train_generator = train_datagen.flow_from_directory('Data',target_size=(100, 100),
batch_size=batch_size,

                                                  class_mode='categorical')


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

```
test_generator = test_datagen.flow_from_directory('Data', target_size=(200, 100),
batch_size=batch_size,

                                        class_mode='categorical',shuffle=False)



import tensorflow as tf


model = tf.keras.models.Sequential([


   # The first convolution

   tf.keras.layers.Conv2D(16, (3,3), activation='relu', input_shape=(100, 100, 3)),

   tf.keras.layers.MaxPooling2D(2, 2),

   # The second convolution

   tf.keras.layers.Conv2D(32, (3,3), activation='relu'),

   tf.keras.layers.MaxPooling2D(2,2),

   # The third convolution

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

   tf.keras.layers.MaxPooling2D(2,2),

   # The fourth convolution

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

   tf.keras.layers.MaxPooling2D(2,2),

   # The fifth convolution

   tf.keras.layers.Conv2D(64, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),

    # Flatten the results to feed into a dense layer

    tf.keras.layers.Flatten(),

    # 128 neuron in the fully-connected layer

    tf.keras.layers.Dense(128, activation='relu'),

    # 5 output neurons for 3 classes with the softmax activation

    tf.keras.layers.Dense(5, activation='softmax')

])


model.summary()


from tensorflow.keras.optimizers import RMSprop

early = tf.keras.callbacks.EarlyStopping(monitor='val_loss',patience=5)

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


total_sample=train_generator.n


n_epochs = 3


history =
model.fit_generator(train_generator,steps_per_epoch=int(total_sample/batch_size),epochs=n
_epochs, verbose=1)
```

```
model.save('model.h5')




acc = history.history['accuracy']


loss = history.history['loss']


epochs = range(1, len(acc) + 1)


# Train and validation accuracy

plt.plot(epochs, acc, 'b', label=' accurarcy')

plt.title('accurarcy')

plt.legend()


plt.figure()


# Train and validation loss

plt.plot(epochs, loss, 'b', label=' loss')

plt.title('  loss')

plt.legend()

plt.show()
```

```
'''from sklearn.metrics import classification_report

from sklearn.metrics import confusion_matrix

test_steps_per_epoch = numpy.math.ceil(test_generator.samples / test_generator.batch_size)


predictions = model.predict_generator(test_generator, steps=test_steps_per_epoch)

# Get most likely class

predicted_classes = numpy.argmax(predictions, axis=1)


true_classes = test_generator.classes

class_labels = list(test_generator.class_indices.keys())


print('Classification Report')



report = classification_report(true_classes, predicted_classes, target_names=class_labels)

print(report)


print('confusion matrix')


confusion_matrix= confusion_matrix(true_classes, predicted_classes)


print(confusion_matrix)
```

```
sns.heatmap(confusion_matrix, annot = True)

plt.show()"
```

APP.PY

```
from flask import Flask, render_template, flash, request, session, send_file

from flask import render_template, redirect, url_for, request

import warnings

import  datetime

import cv2


app = Flask(__name__)

app.config['DEBUG']

app.config['SECRET_KEY'] = '7d441f27d441f27567d441f2b6176a'




@app.route("/")

def homepage():

    return render_template('index.html')




@app.route("/Training")

def Training():

    return render_template('Tranning.html')
```

```
@app.route("/Test")
def Test():
    return render_template('Test.html')



@app.route("/train", methods=['GET', 'POST'])
def train():
    if request.method == 'POST':
        import model as model


        return render_template('Tranning.html')



@app.route("/testimage", methods=['GET', 'POST'])
def testimage():
    if request.method == 'POST':


        file = request.files['fileupload']
        file.save('static/Out/Test.jpg')


        img = cv2.imread('static/Out/Test.jpg')
        if img is None:
            print('no data')
```

```
img1 = cv2.imread('static/Out/Test.jpg')

print(img.shape)

img = cv2.resize(img, ((int)(img.shape[1] / 5), (int)(img.shape[0] / 5)))

original = img.copy()

neworiginal = img.copy()

cv2.imshow('original', img1)

gray = cv2.cvtColor(img1, cv2.COLOR_BGR2GRAY)


img1S = cv2.resize(img1, (960, 540))


cv2.imshow('Original image', img1S)

grayS = cv2.resize(gray, (960, 540))

cv2.imshow('Gray image', grayS)


gry = 'static/Out/gry.jpg'


cv2.imwrite(gry, grayS)

from PIL import ImageOps, Image


im = Image.open(file)


im_invert = ImageOps.invert(im)

inv = 'static/Out/inv.jpg'

im_invert.save(inv, quality=95)
```

```
dst = cv2.fastNlMeansDenoisingColored(img1, None, 10, 10, 7, 21)

cv2.imshow("Nosie Removal", dst)

noi = 'static/Out/noi.jpg'


cv2.imwrite(noi, dst)


import warnings

warnings.filterwarnings('ignore')


import tensorflow as tf

classifierLoad = tf.keras.models.load_model('model.h5')


import numpy as np

from keras.preprocessing import image


test_image = image.load_img('static/Out/Test.jpg', target_size=(100, 100))

img1 = cv2.imread('static/Out/Test.jpg')

# test_image = image.img_to_array(test_image)

test_image = np.expand_dims(test_image, axis=0)

result = classifierLoad.predict(test_image)


out = ''

fer = ''
```

```
        if result[0][0] == 1:


            out = "Arborio"

        elif result[0][1] == 1:

            out = "Basmati"

        elif result[0][2] == 1:

            out = "Ipsala"


        elif result[0][3] == 1:

            out = "Jasmine"




        elif result[0][4] == 1:

            out = "Karacadag"


        org = 'static/Out/Test.jpg'

        gry = 'static/Out/gry.jpg'

        inv = 'static/Out/inv.jpg'

        noi = 'static/Out/noi.jpg'


        return render_template('Test.html', fer=fer, result=out, org=org, gry=gry, inv=inv,
noi=noi)

if _name___== '_main_':

    app.run(debug=True, use_reloader=True)
```

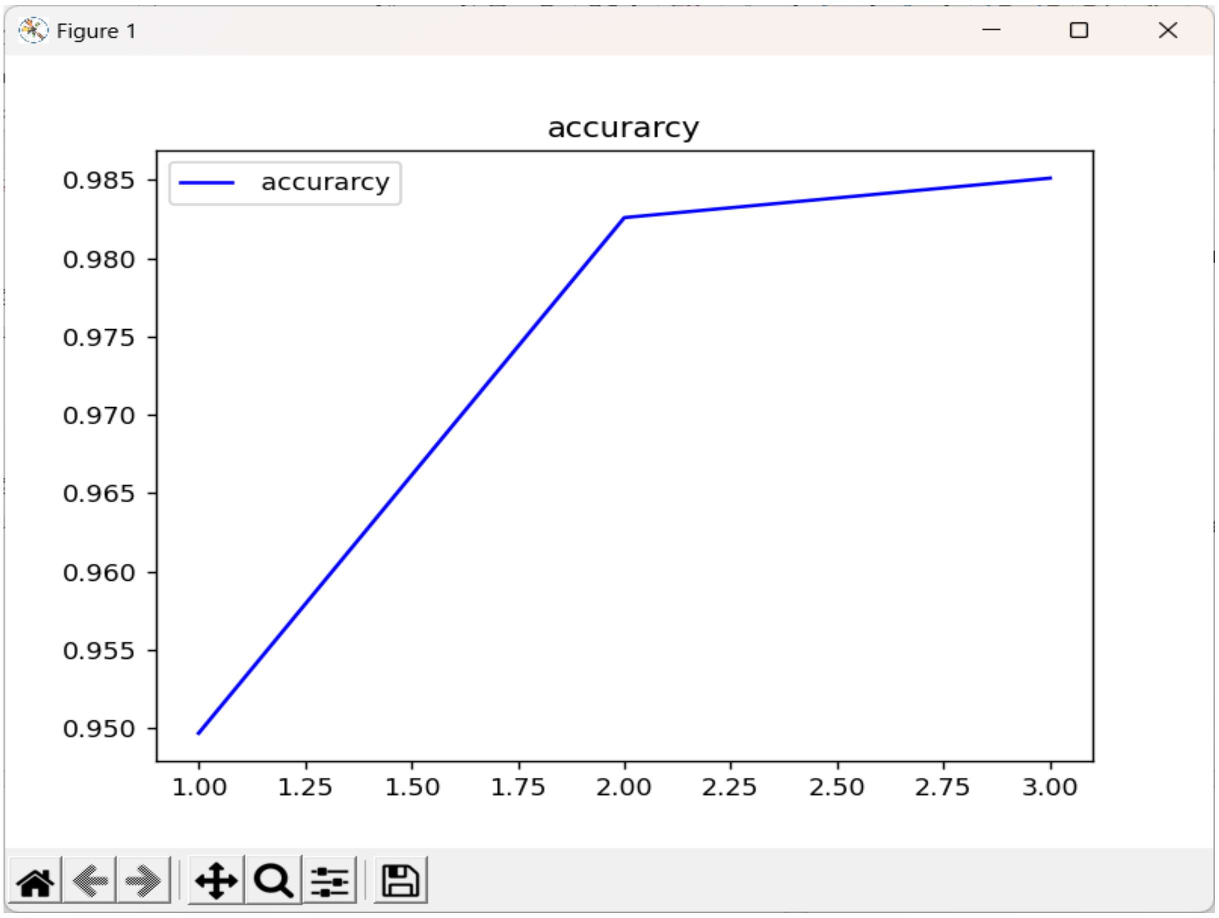
10.2 SCREENSHOTS

MODEL BUILD

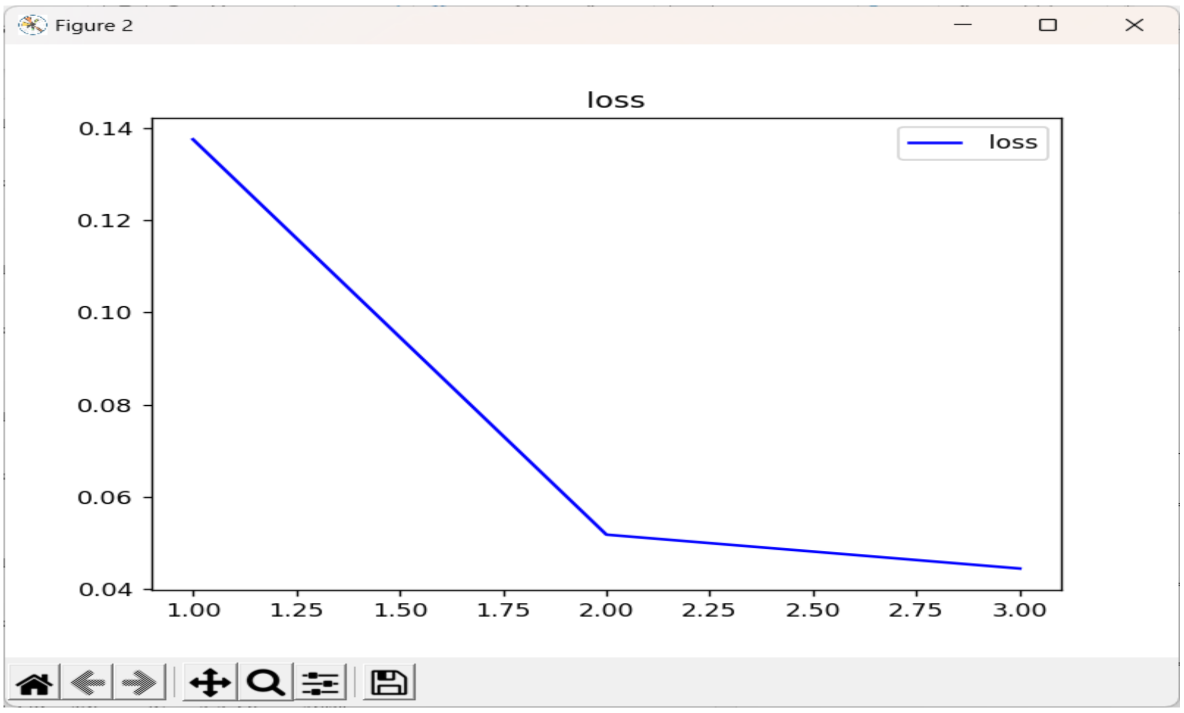
```
E:\project\RiceClassificationPy\venv\Scripts\python.exe
E:/project/RiceClassificationPy/model.py

2024-03-25 21:13:03.352278: W
tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic
library 'cudart64_110.dll'; dlerror: cudart64_110.dll not found

2024-03-25 21:13:03.352609: I tensorflow/stream_executor/cuda/cudart_stub.cc:29] Ignore
above cudart dlerror if you do not have a GPU set up on your machine.

Found 40136 images belonging to 5 classes.

Found 40136 images belonging to 5 classes.

2024-03-25 21:13:10.153365: W
tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic
library 'nvcuda.dll'; dlerror: nvcuda.dll not found

2024-03-25 21:13:10.153476: W tensorflow/stream_executor/cuda/cuda_driver.cc:269] failed
call to cuInit: UNKNOWN ERROR (303)

2024-03-25 21:13:10.156853: I tensorflow/stream_executor/cuda/cuda_diagnostics.cc:169]
retrieving CUDA diagnostic information for host: DESKTOP-V6T9J75

2024-03-25 21:13:10.156992: I tensorflow/stream_executor/cuda/cuda_diagnostics.cc:176]
hostname: DESKTOP-V6T9J75

2024-03-25 21:13:10.159820: I tensorflow/core/platform/cpu_feature_guard.cc:151] This
TensorFlow binary is optimized with oneAPI Deep Neural Network Library (oneDNN) to use
the following CPU instructions in performance-critical operations:  AVX AVX2

To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.

Model: "sequential"

_________________________________________________________________

 Layer (type)                Output Shape              Param #

=================================================================
 conv2d (Conv2D)              (None, 98, 98, 16)        448


 max_pooling2d (MaxPooling2D  (None, 49, 49, 16)        0
 )


 conv2d_1 (Conv2D)            (None, 47, 47, 32)        4640


 max_pooling2d_1 (MaxPooling  (None, 23, 23, 32)        0
 2D)


 conv2d_2 (Conv2D)            (None, 21, 21, 64)        18496


 max_pooling2d_2 (MaxPooling  (None, 10, 10, 64)        0
 2D)


 conv2d_3 (Conv2D)            (None, 8, 8, 64)          36928


 max_pooling2d_3 (MaxPooling  (None, 4, 4, 64)          0
 2D)


 conv2d_4 (Conv2D)            (None, 2, 2, 64)          36928


 max_pooling2d_4 (MaxPooling  (None, 1, 1, 64)          0
 2D)
```

```
 flatten (Flatten)          (None, 64)              0


 dense (Dense)               (None, 128)             8320


 dense_1 (Dense)              (None, 5)               645


=================================================================
Total params: 106,405
Trainable params: 106,405
Non-trainable params: 0
_________________________________________________________________
Epoch 1/3
1254/1254 [==============================] - 287s 229ms/step - loss: 0.1375 -
accuracy: 0.9497
Epoch 2/3
1254/1254 [==============================] - 162s 129ms/step - loss: 0.0518 -
accuracy: 0.9826
Epoch 3/3
1254/1254 [==============================] - 140s 112ms/step - loss: 0.0445 -
accuracy: 0.9851
```

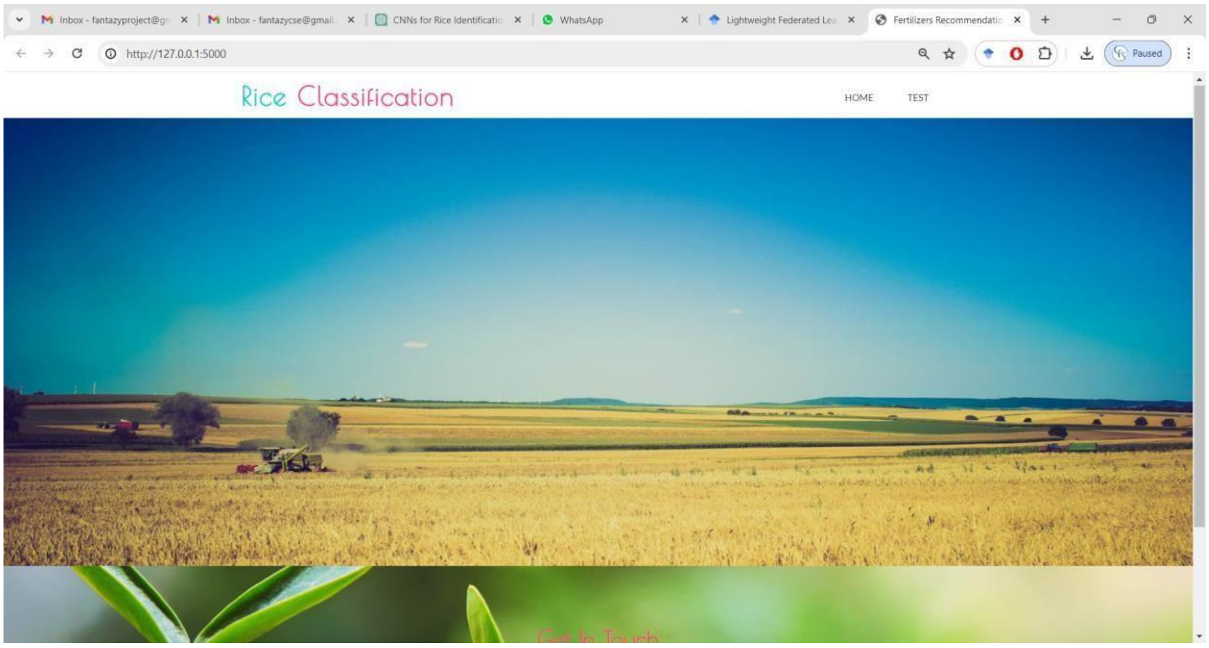
TRAINING ACCURACY

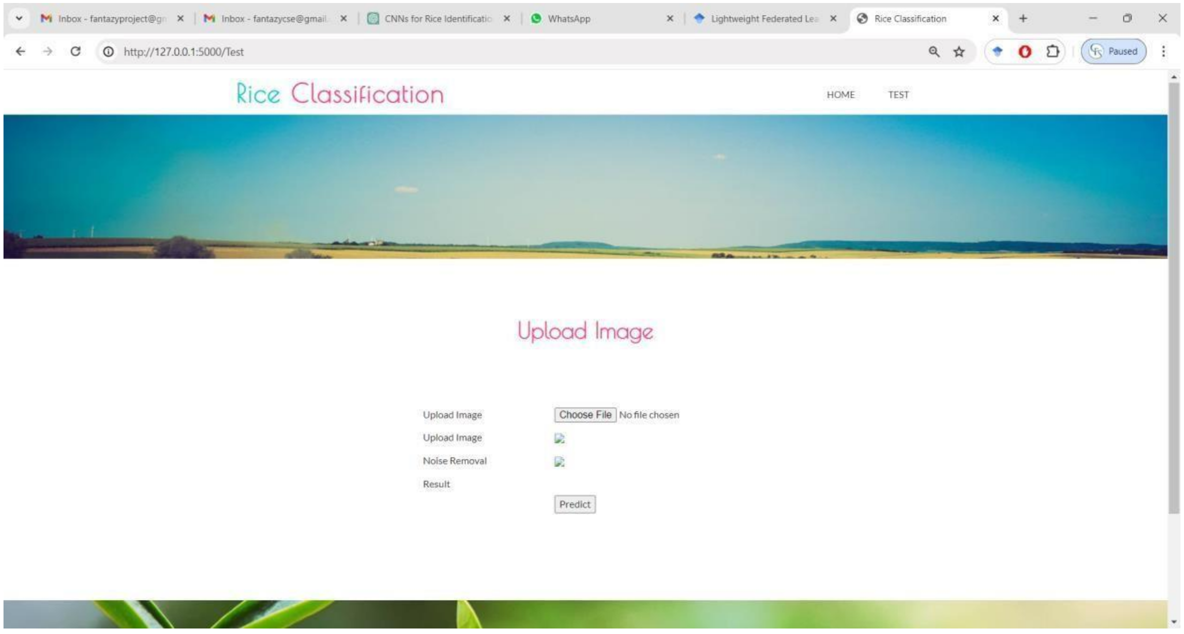


TRAINING LOSS



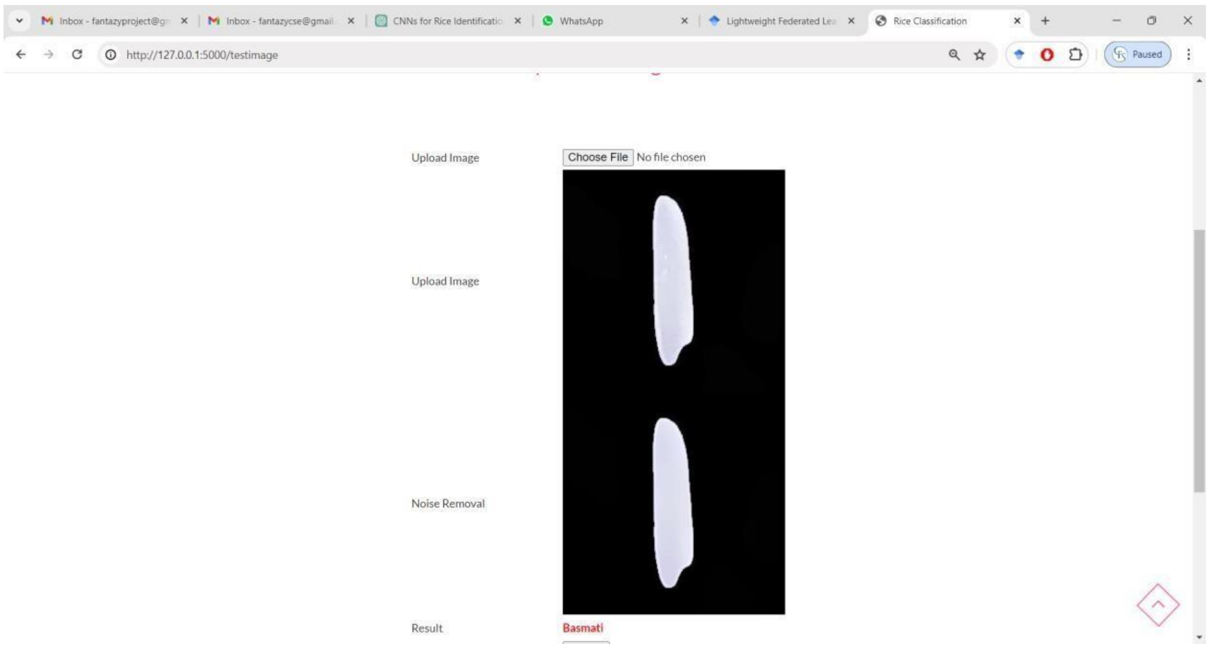
PREDICTION





Upload Image — No file chosen
Upload Image
Noise Removal
Result

NOISE REMOVED IMAGE


Inbox - fantazyproject@gm
Inbox - fantazycse@gmail
CNNs for Rice Identificatio
WhatsApp
Lightweight Federated Lea
Rice Classification
http://127.0.0.1:5000/testimage
Paused

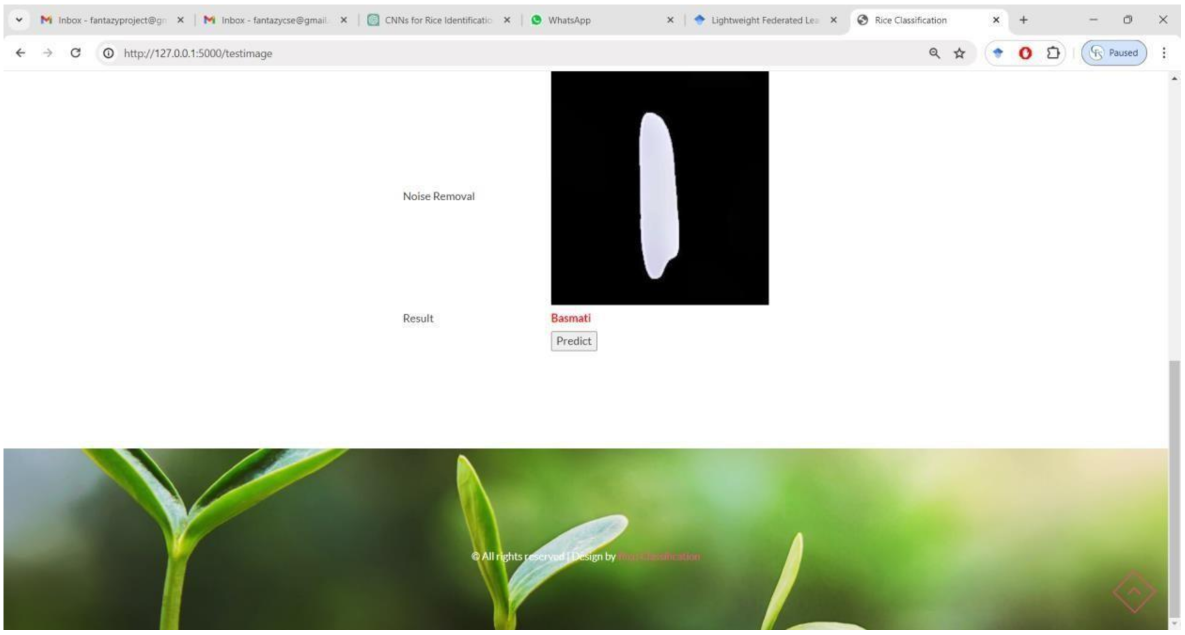
Upload Image
Choose File No file chosen

Upload Image

Noise Removal

Result Basmati

RESULT


Inbox - fantazyproject@gm
Inbox - fantazycse@gmail
CNNs for Rice Identificatio
WhatsApp
Lightweight Federated Lea
Rice Classification
http://127.0.0.1:5000/testimage
Paused

Noise Removal

Result Basmati

Predict

© All rights reserved | Design by Rice Classification

CHAPTER 11

11. CONCLUSION AND FUTURE ENHANCEMENT

11.1 CONCLUSION

In conclusion, the application of deep learning, particularly Convolutional Neural Networks (CNNs), for rice varietal identification represents a significant advancement in agricultural technology. Through the development and implementation of CNN-based classification models, we have demonstrated the capability to accurately distinguish between different rice varieties based on visual characteristics extracted from high-resolution images of rice grains. Our study showcases the effectiveness of CNNs in capturing and learning intricate patterns and features inherent to rice grains, including shape, size, color, and texture. By leveraging transfer learning techniques and fine-tuning pretrained CNN models, we have achieved high classification accuracy, outperformed traditional methods and demonstrated the potential of deep learning for rice varietal identification. This advancement holds promise for improving agricultural practices, enhancing food security, and driving innovation in rice cultivation and research. Further research and development in this area have the potential to revolutionize rice varietal identification and contribute to the sustainable production of this vital crop worldwide.

11.2 FUTURE ENHANCEMENT

Increasing the diversity and size of the training dataset by incorporating images from various sources, environments, and growth stages can enhance the model's ability to generalize across different conditions and improve robustness and continuously refining and optimizing the CNN architecture specifically tailored for rice varietal identification can lead to improved efficiency, speed, and accuracy. Exploring novel architectures or incorporating attention mechanisms may further enhance feature extraction and classification capabilities.

REFERENCES

[1] Tran-Thi-Kim, Nga, et al. "Enhancing the Classification Accuracy of Rice Varieties by Using Convolutional Neural Networks." International Journal of Electrical and Electronic Engineering & Telecommunications 12.2 (2023): 150-160.

[2] Rathnayake, Namal, et al. "Age classification of rice seeds in japan using gradient-boosting and anfis algorithms." Sensors 23.5 (2023): 2828.

[3] Setiawan, Aji, Kusworo Adi, and Catur Edi Widodo. "Rice Foreign Object Classification Based on Integrated Color and Textural Feature Using Machine Learning." Mathematical Modelling of Engineering Problems 10.2 (2023).

[4] Cui, Jiapeng, and Feng Tan. "Rice plaque detection and identification based on an improved convolutional neural network." Agriculture 13.1 (2023): 170.

[5] Aggarwal, Meenakshi, et al. "Lightweight federated learning for rice leaf disease classification using non independent and identically distributed images." Sustainability 15.16 (2023): 12149.

[6] A. Hamza, M. Attique Khan, S.-H. Wang, A. Alqahtani, S. Alsubai, A. Binbusayyis, H. S. Hussein, T. M. Martinetz, and H. Alshazly, ‘‘COVID-19 classification using chest X-ray images: A framework of CNN-LSTM and improved max value moth flame optimization,’’ Frontiers Public Health, vol. 10, Aug. 2022, Art. no. 948205.

[7] R. A. Khurma, H. Alsawalqah, I. Aljarah, M. A. Elaziz, and R. Damaševicius, ‘‘An enhanced evolutionary software defect prediction method using island moth flame optimization,’’ Mathematics, vol. 9, no. 15, p. 1722, Jul. 2021.

[8] M. A. Khan, , ‘‘Human gait analysis: A sequential framework of lightweight deep learning and improved moth-flame optimization algorithm,’’ Comput. Intell. Neurosci., vol. 2022, Jul. 2022, Art. no. 8238375.

[9] B. D. Satoto, D. R. Anamisa, M. Yusuf, M. K. Sophan, S. O. Khairunnisa, and B. Irmawati, ‘‘Rice seed classification using machine learning and deep learning,’’ in Proc. 7th Int. Conf. Informat. Comput. (ICIC), Dec. 2022, pp. 1–7.

[10] M. Tasci, A. Istanbullu, S. Kosunalp, T. Iliev, I. Stoyanov, and I. Beloev, ‘‘An efficient classification of rice variety with quantized neural networks,’’ Electronics, vol. 12, no. 10, p. 2285, May 2023.