

CROP LEAVES DISEASE PREVENTION USING DEEP LEARNING TECHNIQUES

**Submitted in partial fulfilment of the requirements for the award
of**

Master of Science degree in Computer Science

By

RITSIA. B (Reg. No. 42734018)



**DEPARTMENT OF COMPUTER SCIENCE
SCHOOL OF SCIENCE AND HUMANITIES
SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

**Accredited with Grade “A++” by NAAC
JEPPIAAR NAGAR, RAJIV GANDHI SALAI,
CHENNAI - 600 119**



SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

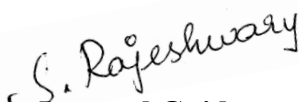
Accredited "A++" Grade by NAAC | 12B Status by UGC | Approved by AICTE

www.sathyabama.ac.in

**DEPARTMENT OF COMPUTER SCIENCE
SCHOOL OF SCIENCE AND HUMANITIES**

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **B. RITSIA (42734018)** who carried out the project entitled “**CROP LEAVES DISEASE PREVENTION USING DEEP LEARNING TECHNIQUES**” under my supervision from January 2024 to April 2024


Internal Guide

Ms. RAJESHWARY.S M.Sc., Assistant Professor



Head of the Department

Dr. REKHA CHAKRAVARTHI M.E., Ph.D.,

Submitted for Viva voce Examination held on

20.04.2024



Internal Examiner



External Examiner

DECLARATION

I am **RITSIA. B (Reg. No. 42734018)** hereby declare that the Project Report entitled "**CROP LEAVES DISEASE PREVENTION USING DEEP LEARNING TECHNIQUES**" done by me under the guidance of **Ms. Rajeshwary.S M.Sc., Assistant Professor**, Department of Computer Science at **SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY**, Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai-600119 is submitted in partial fulfilment of the requirements for the award of Master of Science degree in Computer Science.

DATE: 20.04.2024

RITSIA B

PLACE: CHENNAI

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to Board of Management of **SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. REKHA CHAKRAVARTHI M.E., Ph.D., Dean School of Science and Humanities** for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Ms. Rajeshwary.S M.Sc., Assistant Professor**, Department of computer science for their valuable guidance, suggestions, and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the Department of Computer Science who were helpful in many ways For the Completion of the project.

RITSIA.B

ABSTRACT

Rice is a staple crop and a significant food source for a large portion of the world's population. However, rice crops are susceptible to various diseases that can cause significant yield losses and impact food security. Early detection and accurate diagnosis of rice paddy leaf diseases are crucial for effective disease management and crop protection. We use artificial intelligence techniques for the identification and classification of rice paddy leaf diseases. The first stage of the methodology involves image pre-processing to enhance the quality and extract relevant features from the leaf images. Moreover, the abstract discusses the potential benefits of the developed system for farmers and agricultural experts. By automating the disease detection process, the system can enable early intervention and timely management practices, leading to improved crop yield and reduced economic losses.

TABLE OF CONTENT

CHAPTER NO	TITLE	PAGE NO
	ABSTRACT	
1.	INTRODUCTION	1
1.1	DATA SCIENCE	1
1.2	ARTIFICIAL INTELLIGENCE	2
2.	LITERATURE SURVEY	4
3.	SYSTEM STUDY	5
3.1	OBJECTIVES	5
3.2	PROJECT GOAL	6
3.3	SCOPE OF THE PROJECT	6
4.	METHODOLOGIES	7
4.1	EXISTING SYSTEM	7
4.2	PROPOSED SYSTEM	8
5.	PROJECT REQUIREMENTS	9
5.1	HARDWARE REQUIREMENTS	9
5.2	SOFTWARE REQUIREMENTS	9
6.	FEASIBILITY STUDY	16

7.	SYSTEM ARCHITECTURE	20
8.	WORKFLOW DIAGRAM	21
9.	USECASE DIAGRAM	22
10.	CLASS DIAGRAM	23
11.	ACTIVITY DIAGRAM	24
12.	SEQUENCE DIAGRAM	25
13.	ER – DIAGRAM	26
14.	CONCLUSION AND FUTURE ENHANCEMENT	35
14.1	CONCLUSION	35
14.2	FUTURE ENHANCEMENT	35
	REFERENCES	36
15.	APPENDIX	37
A.	SOURCE CODE	37

LIST OF FIGURES

FIGURE NO	FIGURE NAME	PAGE NO
1.	SYSTEM ARCHITECTURE	20
2.	WORKFLOW DIAGRAM	21
3.	USE-CASE DIAGRAM	22
4.	CLASS DIAGRAM	23
5.	ACTIVITY DIAGRAM	24
6.	SEQUENCE DIAGRAM	25
7.	ER – DIAGRAM	26

CHAPTER 1

1.INTRODUCTION

In today's dynamic financial landscape, accurate stock price prediction is paramount for investors, traders, and financial institutions. This study focuses on forecasting stock prices for two technology giants, Amazon, and Google, utilizing advanced machine learning techniques and sentiment analysis of news articles. We analyse historical stock price data, extract key financial indicators, and employ diverse machine learning algorithms, including regression and deep learning models, to create predictive models that capture complex price dynamics. Additionally, we integrate sentiment analysis of news articles to gauge market sentiment's impact on stock prices. Our evaluation, using metrics like MAE, RMSE, and accuracy, sheds light on model effectiveness and underscores the significance of sentiment analysis in predicting stock prices for technology leaders like Amazon and Google, offering valuable insights for investors and financial professionals operating in this fast-paced sector.

1.1 DATA SCIENCE

Data science is an interdisciplinary field that uses scientific methods, processes, algorithms, and systems to extract knowledge and insights from structured and unstructured data, and apply knowledge and actionable insights from data across a broad range of application domains.

The term "data science" has been traced back to 1974, when Peter Naur proposed it as an alternative name for computer science.

The term “data science” was first coined in 2008 by D.J. Patil, and Jeff Hammerbacher, the pioneer leads of data and analytics efforts at LinkedIn and Facebook.

Required Skills for a Data Scientist:

- **Programming:** Python, SQL, Scala, Java, R, MATLAB.
- **Machine Learning:** Natural Language Processing, Classification, Clustering.
- **Data Visualization:** Tableau, SAS, D3.js, Python, Java, R libraries.
- **Big data platforms:** MongoDB, Oracle, Microsoft Azure, Cloudera.

1.2 ARTIFICIAL INTELLIGENCE

Artificial intelligence (AI) refers to the simulation of human intelligence in machines that are programmed to think like humans and mimic their actions. The term may also be applied to any machine that exhibits traits associated with a human mind such as learning and problem-solving.

Artificial intelligence (AI) is [intelligence](#) demonstrated by [machines](#), as opposed to the natural intelligence [displayed by humans](#) or [animals](#). Leading AI textbooks define the field as the study of "[intelligent agents](#)" any system that perceives its environment and takes actions that maximize its chance of achieving its goals. Some popular accounts use the term "artificial intelligence" to describe machines that mimic "cognitive" functions that humans associate with the [human mind](#), such as "learning" and "problem solving", however this definition is rejected by major AI researchers.

Learning processes. This aspect of AI programming focuses on acquiring data and creating rules for how to turn the data into actionable information. The

rules, which are called algorithms, provide computing devices with step-by-step instructions for how to complete a specific task.

Reasoning processes. This aspect of AI programming focuses on choosing the right algorithm to reach a desired outcome.

Self-correction processes. This aspect of AI programming is designed to continually fine-tune algorithms and ensure they provide the most accurate results possible.

Natural Language Processing (NLP):

[Natural language processing](#) (NLP) allows machines to read and [understand](#) human language. A sufficiently powerful natural language processing system would enable [natural-language user interfaces](#) and the acquisition of knowledge directly from human-written sources, such as newswire texts. Some straightforward applications of natural language processing include [information retrieval](#), [text mining](#), [question answering](#) and [machine translation](#). Many current approaches use word co-occurrence frequencies to construct syntactic representations of text. "Keyword spotting" strategies for search are popular and scalable but dumb; a search query for "dog" might only match documents with the literal word "dog" and miss a document with the word "poodle". "Lexical affinity" strategies use the occurrence of words such as "accident" to [assess the sentiment](#) of a document.. Beyond semantic NLP, the ultimate goal of "narrative" NLP is to embody a full understanding of common sense reasoning. By 2019, [transformer](#)-based deep learning architectures could generate coherent text.

CHAPTER 2

2. LITERATURE SURVEY

YEAR	TITLE	AUTHOR	DESCRIPTION
2021	Rice Leaf Diseases Recognition Using Convolutional Neural Networks	Syed Md. Minhaz Hossain ¹ , Md. Monjur Morhsed	Developing an accurate and efficient rice leaf diseases recognition system employing Convolutional Neural Networks for automated plant health monitoring.
2021	Paddy Leaf Disease Detection using CNN	Nageswara Rao Moparthy ¹ , Debnath Bhattacharyya	Utilizing Convolutional Neural Networks (CNN), this model accurately identifies and classifies paddy leaf diseases based on intricate image features, enhancing precision in agricultural disease detection.
2022	Rice Leaf Disease Detection and Diagnosis Using Convolution Neural Network	Panchami R	Developing a robust Convolutional Neural Network for precise detection and diagnosis of rice leaf diseases by analyzing intricate patterns in leaf images.
2022	Deep Learning Utilization in Agriculture: Detection of Rice Plant Diseases Using an Improved CNN Model	Ghazanfar Latif ^{1,2,*} , Sherif E. Abdelhamid ³	This study employs an enhanced Convolutional Neural Network (CNN) model for the precise detection of rice plant diseases, showcasing the potential of deep learning in revolutionizing agricultural diagnostics.
2022	RICE PLANT DISEASE IDENTIFICATION DECISION SUPPORT MODEL USING MACHINE LEARNING	Gaurav Shrivastava and Harish Patidar	A machine learning-based decision support model for identifying rice plant diseases, enhancing agricultural productivity through efficient disease diagnosis.
2023	PREDICTING THE RICE LEAF DISEASES USING CNN	Ms. M. Anitha, Mr. E. Nagaraju	Utilizing Convolutional Neural Networks (CNN) for predicting rice leaf diseases involves training deep learning models on image data to discern and classify various diseases based on leaf characteristics.

CHAPTER 3

3.SYSTEM STUDY

3.1 Aim:

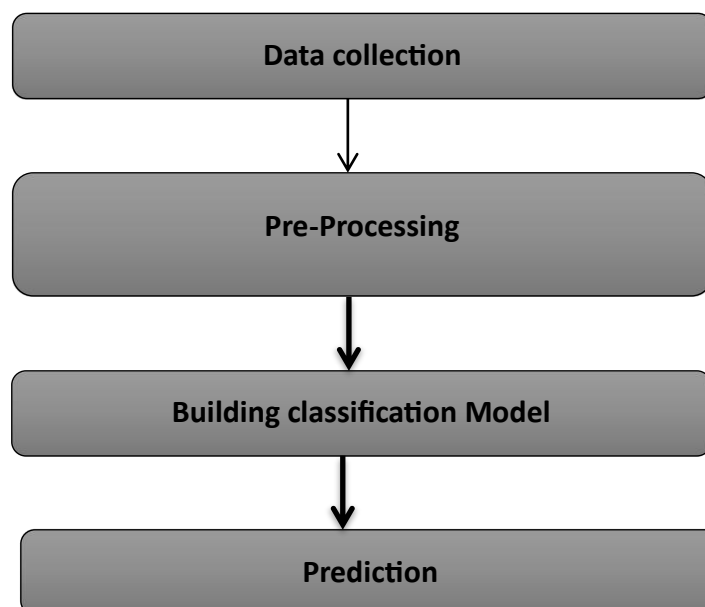
To design such system that can detect crop disease accurately. Create database of insecticides for respective disease. To provide remedy for the disease that is detected. So, we cannot able to classify it easily. So, this project can easily classify the paddy leaves disease.

Objectives:

The goal is to develop a deep learning model for paddy leaves disease image classification by convolutional neural network algorithm for potentially classifying the results in the form of best accuracy by comparing the CNN architectures.

Workflow:

The steps involved in Building the data model is depicted below.



3.2 Project Goals

- Exploration data analysis of variable identification
 - Loading the given dataset
 - Import required libraries packages

- Analyse the general properties
- Uni-variate data analysis
 - Rename, add data, and drop the data
 - To specify data type ○ Exploration data analysis
 - Plot images.
- Method of feature engineering
 - Pre-processing the given dataset
 - Splitting the test and training dataset
 - Comparing the Architectures for future improvement
- Comparing architectures to predict the result
 - Based on the best accuracy

3.3 SCOPE OF THE PROJECT

A paddy leaves images are collected. We have to train the machine to classify the types of leaves. This project contains different type's leaves like Bread. We train to teach the machine to achieve the accuracy and get the possible outcome.

CHAPTER 4

4. METHODOLOGIES

4.1 EXISTING SYSTEM:

With the development of plant phonemics, the identification of plant diseases from leaf images has become an effective and economic approach in plant disease science. Among the methods of plant diseases identification, the convolutional neural network (CNN) is the most popular one for its superior performance. In this work, we propose a new method, namely Pit Lid, based on pertained Inception-V3 convolutional neural network and transfer learning to identify plant leaf diseases from phenotype data of plant leaf with small sample size. To evaluate the robustness of the proposed method, the experiments on several datasets with small-scale samples were implemented. This study provides a plant disease identification tool based on a deep learning algorithm for plant phonemics.

Disadvantages:

- They are using more computational consuming architecture.
- They did not use deployment process.
- They did not build their own architecture.
- They have only classified 4 diseases.
- They are not classifying any paddy leaf disease.
- It is very complicated to process.
- They are not implementing CNN Algorithm.

4.2 PROPOSED SYSTEM:

The proposed system aims to utilize Convolution Neural Network (CNNs) for the detection and classification of Rice leaves diseases. CNNs has proven to be effective in analysing medical images and can be leverage to improve the accuracy and efficiency of Rice leave disease. By leveraging the power of deep learning and CNNs, the system can assist research to identify the microorganisms

and aiding in treatment planning and monitoring. Train the model with hyper parameter tuning and additional considerations may be required to develop a crucial system for identifying disease classification using CNN and give the fact about the rice leaves state to the farm professionals for further treatment and precautions.

Advantages:

- It is essential to detect a particular disease of paddy. In our country many farmers are not so educated to get correct information about the diseases. So, it helps a lot
- We are using CNN Algorithm for getting better accuracy.
- Deployment can be implemented.
- We done a production level application using for deploy purpose.
- We classify more than 10 diseases.
- We have built our own architecture for model building.
- We compared more than two architectures to get better accuracy level.

CHAPTER 5

5. PROJECT REQUIREMENTS

5.1 SOFTWARE REQUIREMENTS

1. Software Requirements:

Operating System : Windows 11

Tool : Anaconda with Jupyter Notebook

5.2 HARDWARE REQUIREMENTS

2. Hardware requirements:

Processor : Intel i3

Hard disk : minimum 10 GB

RAM : minimum 4 GB

LIST OF MODULES:

- Manual architecture
- Implementing Architecture 1 (ALEXNET)
- Implementing Architecture 2 (RESNET)
- Deployment

ALEXNET ARCHITECTURE:

AlexNet is a deep convolutional neural network architecture that gained significant attention and played a pivotal role in advancing the field of deep learning and computer vision. It was developed by Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, and won the 2012 ImageNet Large Scale Visual Recognition Challenge (ILSVRC).

Input Layer: The network takes a color image as input. AlexNet was designed for images with a resolution of 224x224 pixels and three color channels (RGB).

Convolutional Layers: AlexNet contains five convolutional layers, each followed by a max-pooling layer. These layers are responsible for learning and extracting various features from the input image.

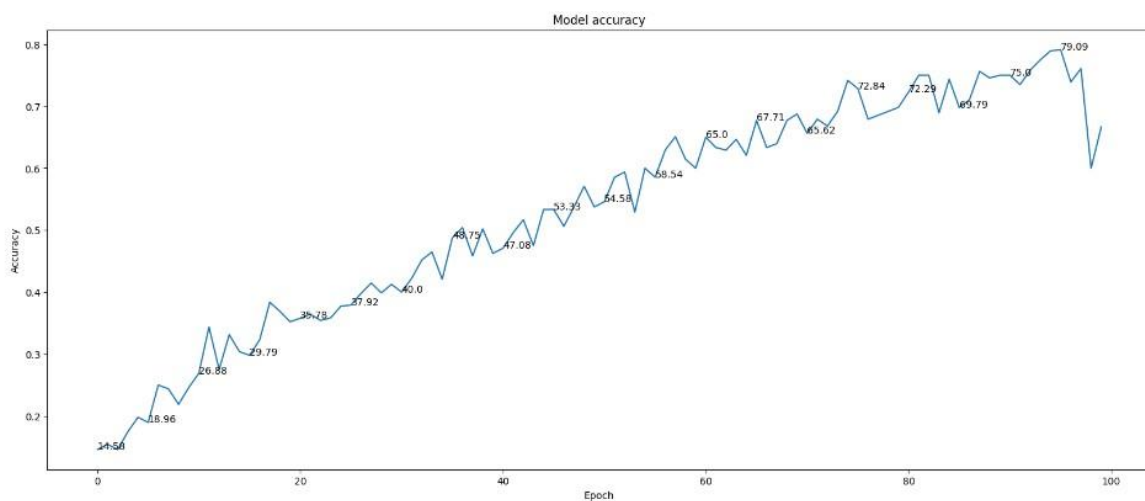
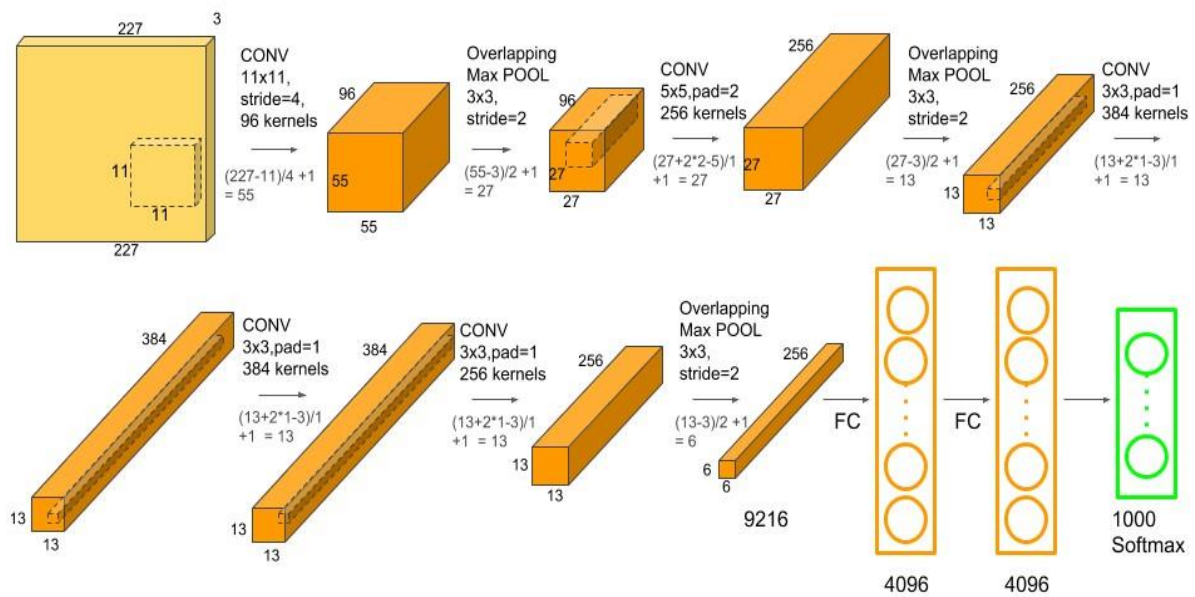
Max-Pooling Layers: Max-pooling is applied after each of the first two convolutional layers. It helps reduce the spatial dimensions and keep the most important features.

Normalization Layers: Local response normalization is applied after the first and second convolutional layers. This helps with generalization.

Fully Connected Layers: After the convolutional and pooling layers, there are three fully connected layers. These layers are traditional feedforward neural network layers.

Output Layer: The output layer has 1000 neurons, each representing a class in the ImageNet dataset. It uses a softmax activation function to compute the class probabilities.

Dropout: Dropout is applied to the two fully connected layers to prevent overfitting.



RESNET ARCHITECTURE:

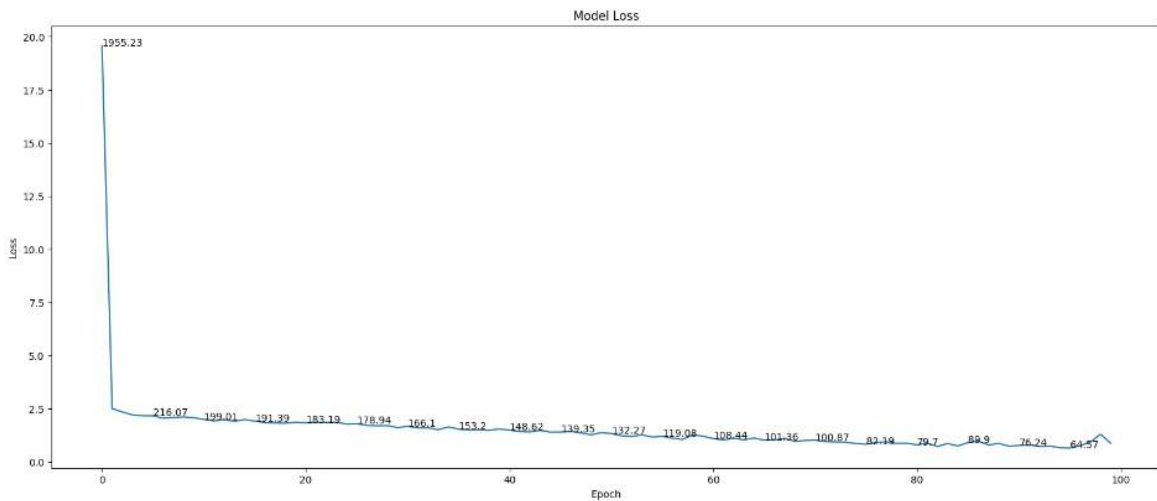
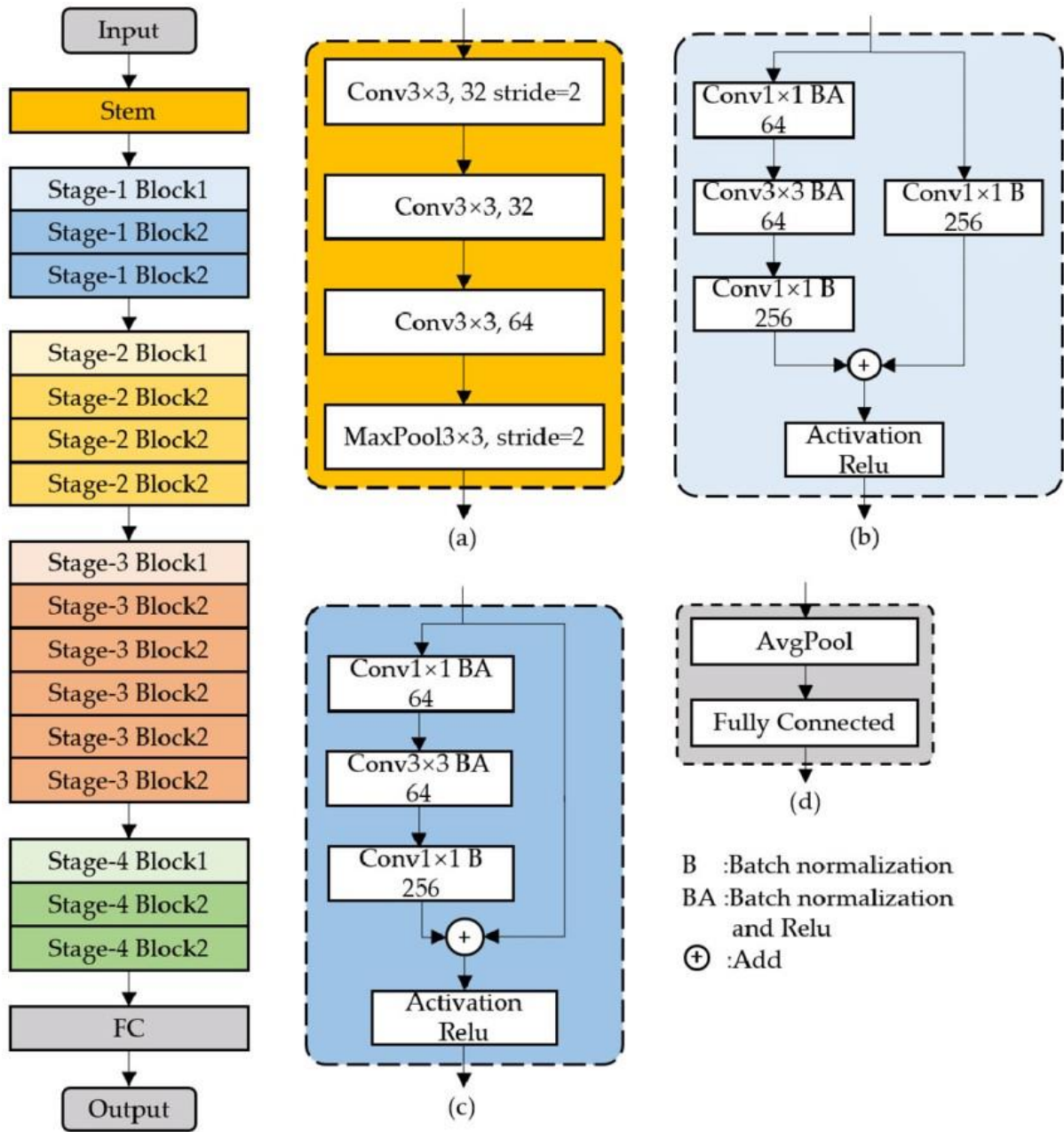
ResNet, short for Residual Network, is a deep convolutional neural network architecture that was introduced by Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun in their 2015 paper titled "Deep Residual Learning for Image Recognition".

Residual Block: The core innovation in ResNet is the residual block. Instead of trying to learn the desired underlying mapping ($H(x)$) directly, ResNet learns the residual mapping ($F(x) = H(x) - x$). The network is forced to learn the residual, which is easier, and then adds it back to the input, effectively "skipping" some layers. This helps mitigate the vanishing gradient problem and makes it easier to train very deep networks.

Skip Connections: Skip connections, also known as shortcut connections or identity mappings, are the fundamental architectural element in ResNet. These connections allow the network to skip one or more layers and pass the input directly to a deeper layer in the network. Skip connections enable the training of extremely deep networks while maintaining good performance.

Bottleneck Architecture: ResNet architectures often use a bottleneck design in the residual blocks to reduce the computational cost. In a bottleneck block, 1×1 , 3×3 , and 1×1 convolutions are used to transform the input, reducing the number of parameters and computational complexity.

Global Average Pooling: Instead of using fully connected layers at the end of the network, ResNet typically employs global average pooling, which computes the average of feature maps over spatial dimensions. This reduces overfitting and decreases the number of parameters.



SOFTWARE DESCRIPTION

Anaconda is a [free and open-source](#) distribution of the [Python](#) and [R](#) programming languages for [scientific computing](#) ([data science](#), [machine learning](#) applications, large-scale data processing, [predictive analytics](#), etc.), that aims to simplify [package management](#) and deployment.

Package versions are managed by the [package management system](#) “Conda”. The Anaconda distribution is used by over 12 million users and includes more than 1400 popular data-science packages suitable for Windows, Linux, and MacOS. So, Anaconda distribution comes with more than 1,400 packages as well as the [Conda](#) package and virtual environment manager called Anaconda Navigator and it eliminates the need to learn to install each library independently. The open-source packages can be individually installed from the Anaconda repository with the conda install command or using the pip install command that is installed with Anaconda. [Pip packages](#) provide many of the features of conda packages and in most cases they can work together. Custom packages can be made using the command, and can be shared `conda build` with others by uploading them to Anaconda Cloud, [PyPI](#) or other repositories. The default installation of Anaconda2 includes Python 2.7 and Anaconda3 includes Python 3.7. However, you can create new environments that include any version of Python packaged with conda.

JUPYTER NOTEBOOK

This website acts as “meta” documentation for the Jupyter ecosystem. It has a collection of resources to navigate the tools and communities in this ecosystem, and to help you get started.

Project Jupyter is a project and community whose goal is to "develop opensource software, open-standards, and services for interactive computing across dozens of programming languages". It was spun off from IPython in 2014 by Fernando Perez.

Notebook documents are documents produced by the Jupyter Notebook App, which contain both computer code (e.g. python) and rich text elements (paragraph, equations, figures, links, etc...). Notebook documents are both human-readable documents containing the analysis description and the results (figures, tables, etc.) as well as executable documents which can be run to perform data analysis.

CHAPTER 6

6. Feasibility study

Data Wrangling

In this section of the report will load in the data, check for cleanliness, and then trim and clean given dataset for analysis. Make sure that the document steps carefully and justify for cleaning decisions.

Data collection

The data set collected for predicting given data is split into Training set and Test set. Generally, 7:3 ratios are applied to split the Training set and Test set. The Data Model which was created using Random Forest, logistic, Decision tree algorithms and Support vector classifier (SVC) are applied on the Training set and based on the test result accuracy, Test set prediction is done.

Preprocessing

The data which was collected might contain missing values that may lead to inconsistency. To gain better results data need to be preprocessed so as to improve the efficiency of the algorithm. The outliers have to be removed and also variable conversion need to be done.

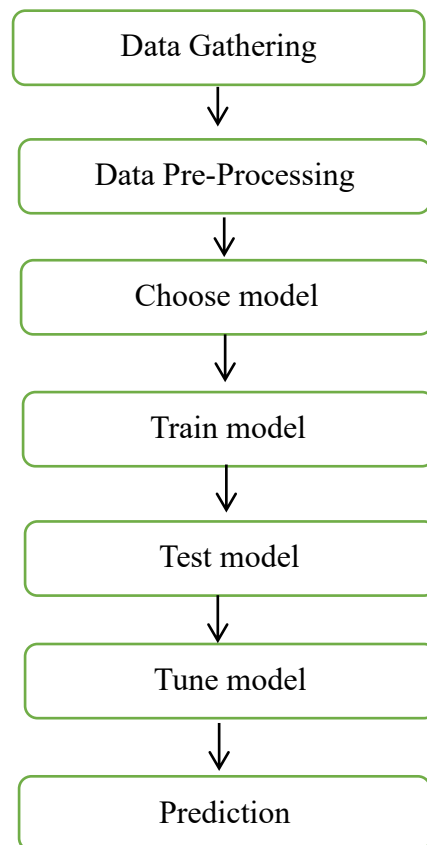
Building the classification model

The prediction of Human Air Pollution, a high accuracy prediction model is effective because of the following reasons: It provides better results in classification problem.

- It is strong in preprocessing outliers, irrelevant variables, and a mix of continuous, categorical, and discrete variables.
- It produces out of bag estimate error which has proven to be unbiased in many tests and it is relatively easy to tune with.

Construction of a Predictive Model

Machine learning needs data gathering have lot of past data. Data gathering have sufficient historical data and raw data. Before data pre-processing, raw data cannot be used directly. It is used to pre-process then, what kind of algorithm with model. Training and testing this model working and predicting correctly with minimum errors. Tuned model involved by tuned time to time with improving the accuracy.



Process of dataflow diagram

MACHINE LEARNING

Machine learning is to predict the future from past data. Machine learning (ML) is a type of artificial intelligence (AI) that provides computers with the ability to learn without being explicitly programmed. Machine learning focuses on the development of Computer Programs that can change when exposed to new data and the basics of Machine Learning, implementation of a simple machine learning algorithm using python. Process of training and prediction involves use of specialized algorithms. It feed the training data to an algorithm, and the algorithm uses this training data to give predictions on a new test data. Machine learning can be roughly separated in to three categories. There are supervised learning, unsupervised learning and reinforcement learning. Supervised learning program is both given the input data and the corresponding labelling to learn data has to be labelled by a human being beforehand. Unsupervised learning is no labels. It provided to the learning algorithm. This algorithm has to figure out the clustering of the input data. Finally, Reinforcement learning dynamically interacts with its environment and it receives positive or negative feedback to improve its performance.

Data scientists use many different kinds of machine learning algorithms to discover patterns in python that lead to actionable insights. At a high level, these different algorithms can be classified into two groups based on the way they “learn” about data to make predictions: supervised and unsupervised learning. Classification is the process of predicting the class of given data points. Classes are sometimes called as targets/ labels or categories. Classification predictive modeling is the task of approximating a mapping function from input variables(X) to discrete output variables(y). In machine learning and statistics, classification is a supervised learning approach in which the computer program learns from the data input given to it and then uses this learning to classify new observation. This data set may simply be bi-class (like identifying whether the person is male or female or that the mail is spam or non-spam) or it may be multi-class too. Some

examples of classification problems are: speech recognition, handwriting recognition, bio metric identification, document classification etc.



Supervised Machine Learning is the majority of practical machine learning uses supervised learning. Supervised learning is where have input variables (X) and an output variable (y) and use an algorithm to learn the mapping function from the input to the output is $y = f(X)$. The goal is to approximate the mapping function so well that when you have new input data (X) that you can predict the output variables (y) for that data. Techniques of Supervised Machine Learning algorithms include **logistic regression**, **multi-class classification**, **Decision Trees** and **support vector machines** etc. Supervised learning requires that the data used to train the algorithm is already labelled with correct answers. Supervised learning problems can be further grouped into **Classification** problems. This problem has as goal the construction of a succinct model that can predict the value of the dependent attribute from the attribute variables. The difference between the two tasks is the fact that the dependent attribute is numerical for categorical for classification. A classification model attempts to draw some conclusion from observed values. Given one or more inputs a classification model will try to predict the value of one or more outcomes. A classification problem is when the output variable is a category, such as “red” or “blue”.

PREPARING DATASET:

This dataset is collected from a website which contains image records of features extracted, which were then split into train and test data.

TRAINING DATA FOR bacterial_leaf_blight:

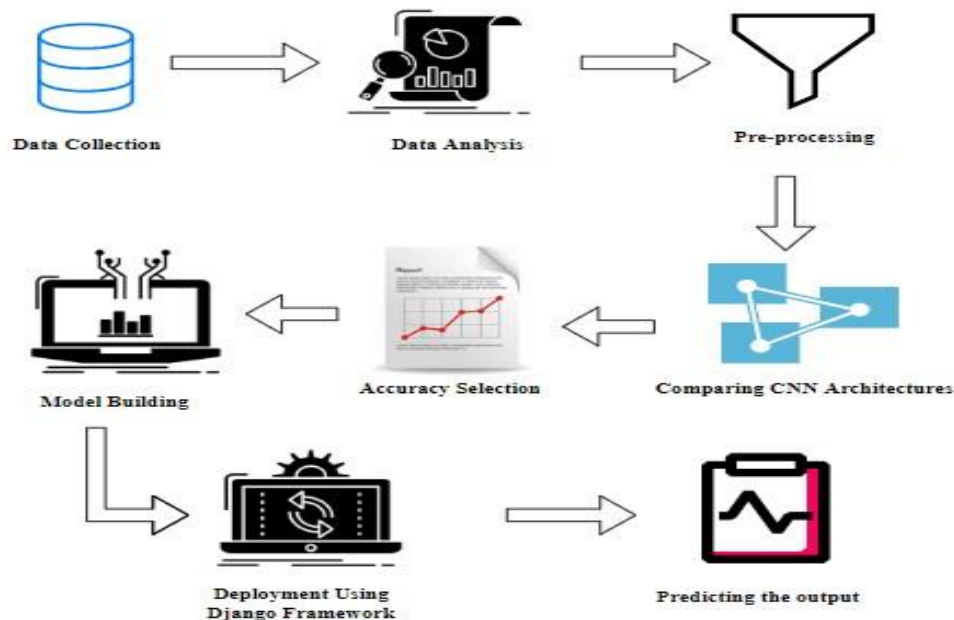
```
===== Images in: DATASET/TRAIN/bacterial_leaf_blight
Images_count : 200
Min_width : 480
Max_width : 480
Min_height : 640
Max_height : 640
```



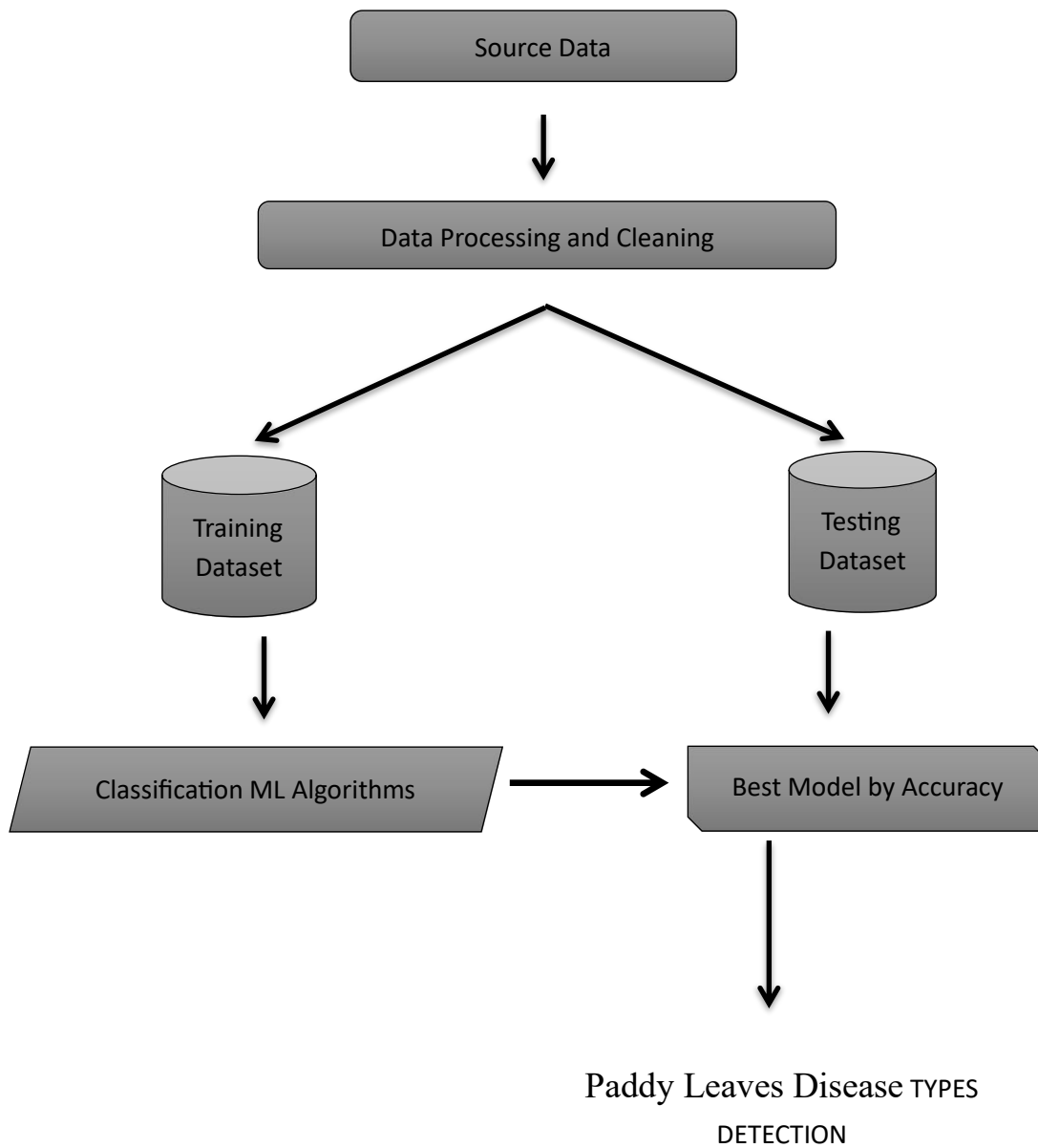
```
1 print("")
2 print("TRAINING DATA FOR bacterial_panicle_blight:")
3 print("")
4 images_details(bacterial_panicle_blight)
5 print("")
6 plot_images(bacterial_panicle_blight, 10)
```

Python

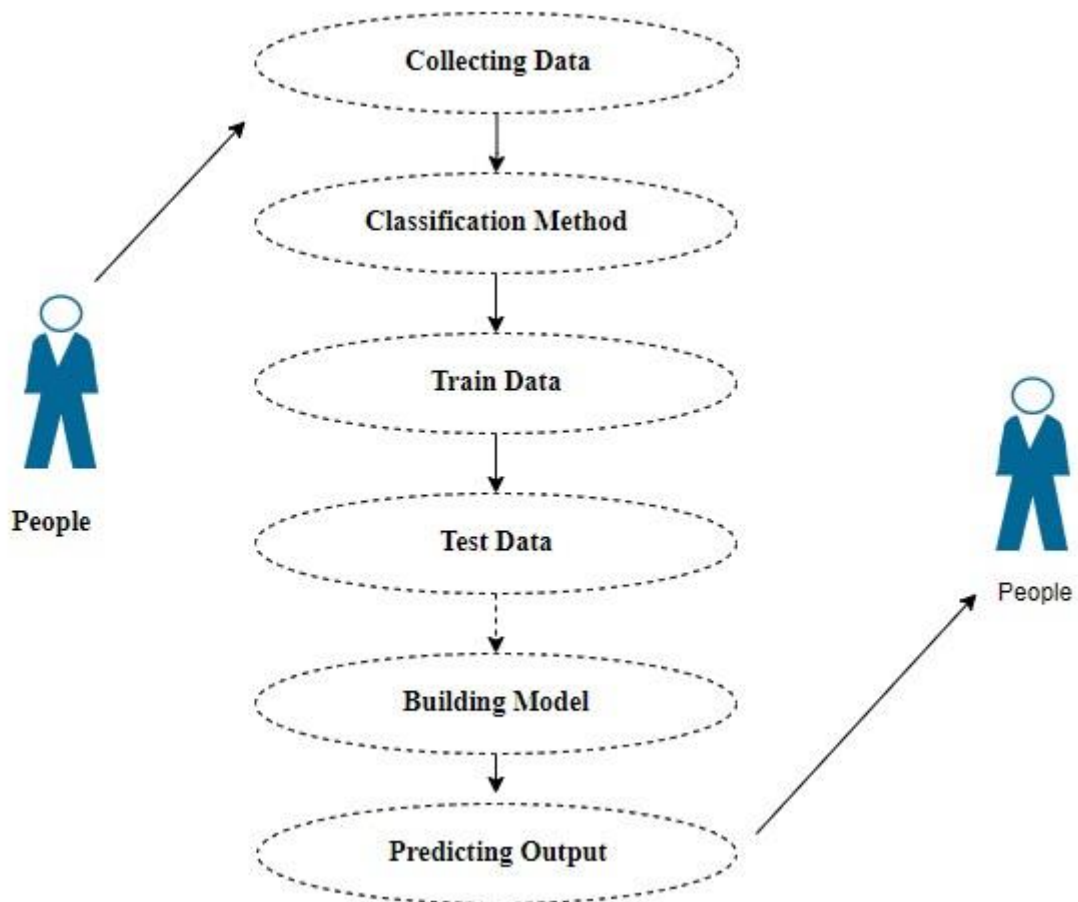
7. SYSTEM ARCHITECTURE



8.WORK FLOW DIAGRAM

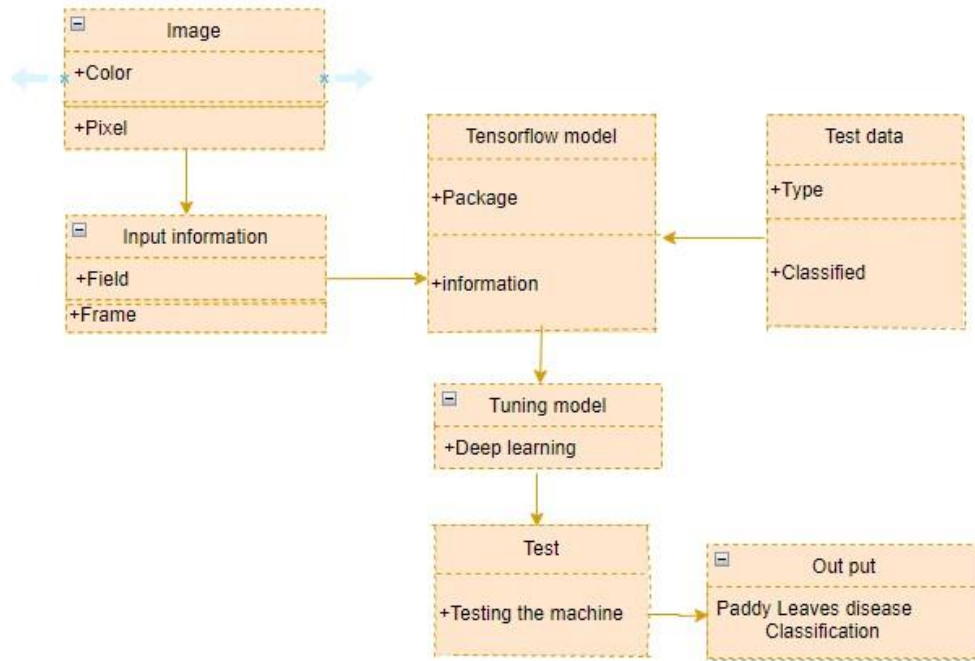


9. Use Case Diagram:



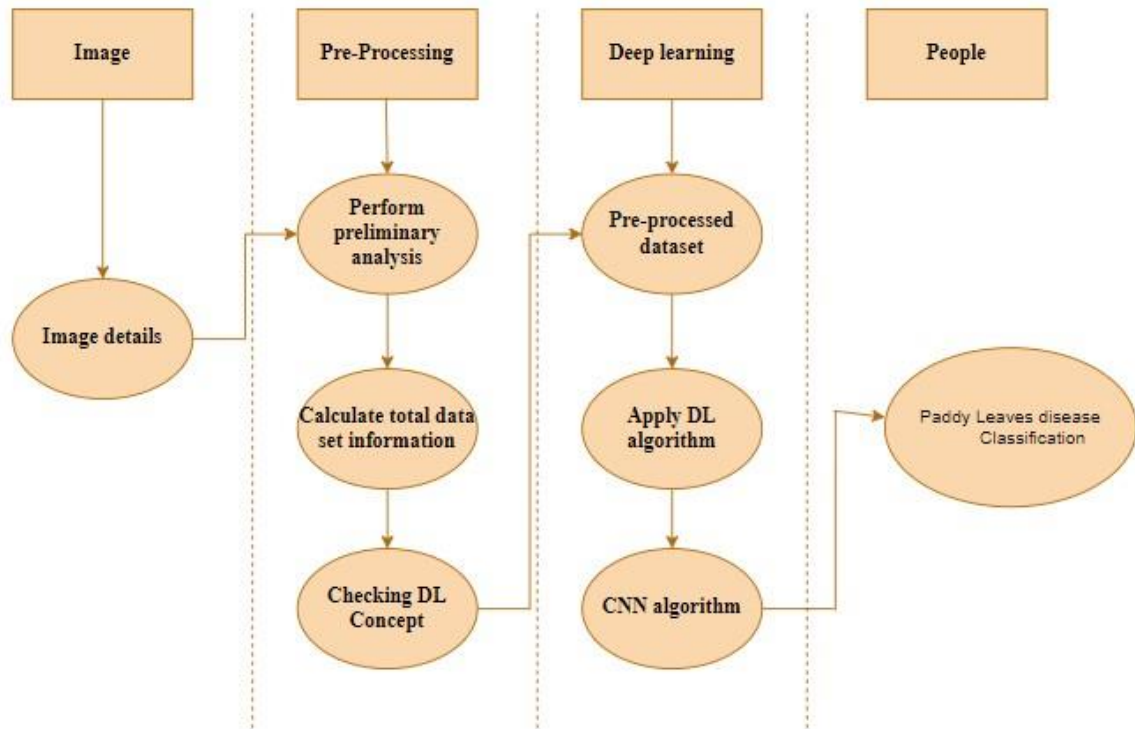
Use case diagrams are considered for high level requirement analysis of a system. So, when the requirements of a system are analysed, the functionalities are captured in use cases. So, it can say that uses cases are nothing but the system functionalities written in an organized manner.

10. Class Diagram:



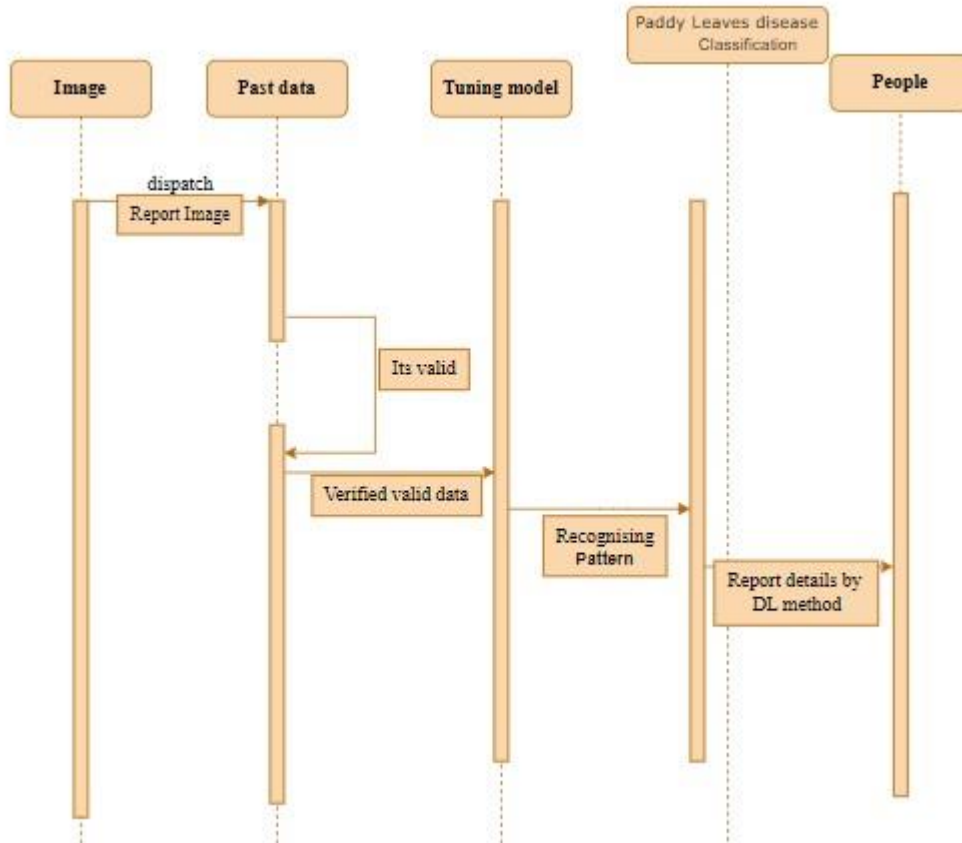
Class diagram is basically a graphical representation of the static view of the system and represents different aspects of the application. So, a collection of class diagrams represents the whole system. The name of the class diagram should be meaningful to describe the aspect of the system. Each element and their relationships should be identified in advance Responsibility (attributes and methods) of each class should be clearly identified for each class minimum number of properties should be specified and because, unnecessary properties will make the diagram complicated. Use notes whenever required to describe some aspect of the diagram and at the end of the drawing it should be understandable to the developer/coder.

11. Activity Diagram:



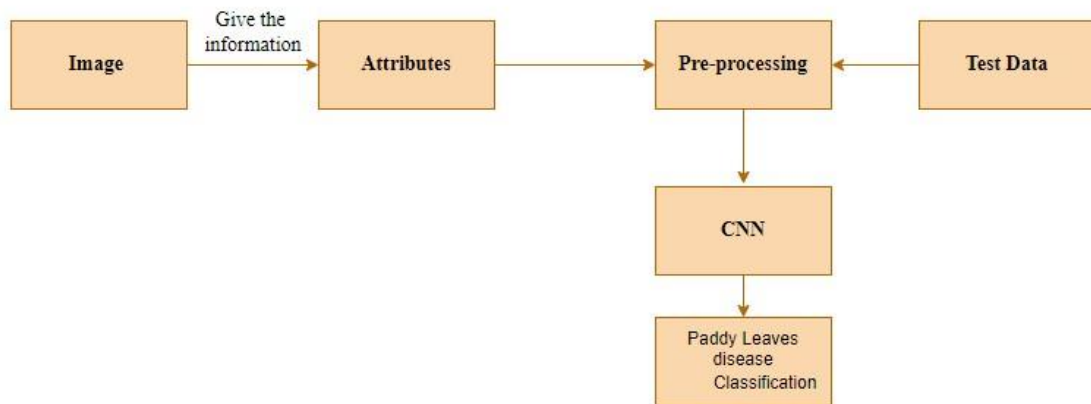
Activity is a particular operation of the system. Activity diagrams are not only used for visualizing dynamic nature of a system but they are also used to construct the executable system by using forward and reverse engineering techniques. The only missing thing in activity diagram is the message part. It does not show any message flow from one activity to another. Activity diagram is some time considered as the flow chart. Although the diagrams looks like a flow chart but it is not. It shows different flow like parallel, branched, concurrent and single.

12. Sequence Diagram:



Sequence diagrams model the flow of logic within your system in a visual manner, enabling you both to document and validate your logic, and are commonly used for both analysis and design purposes. Sequence diagrams are the most popular UML artefact for dynamic modelling, which focuses on identifying the behaviour within your system. Other dynamic modelling techniques include [activity diagramming](#), [communication diagramming](#), [timing diagramming](#), and [interaction overview diagramming](#). Sequence diagrams, along with [class diagrams](#) and [physical data models](#) are in my opinion the most important design-level models for modern business application development.

13. COLLABORATION DIAGRAM:



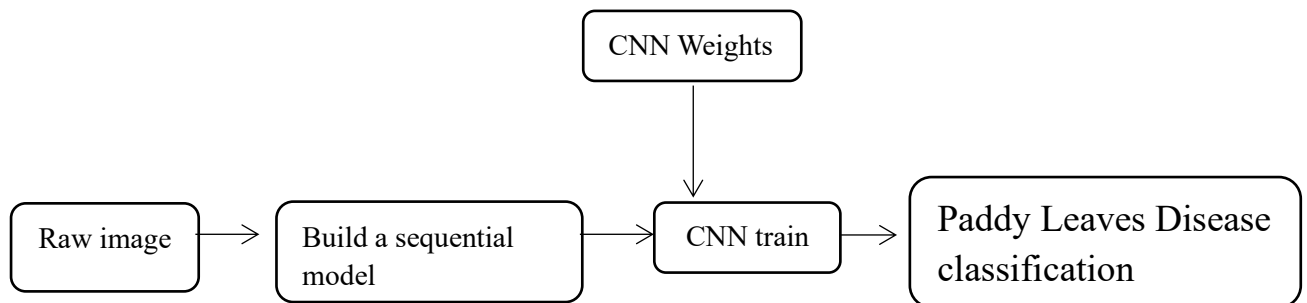
A collaboration diagram show the objects and relationships involved in an interaction, and the sequence of messages exchanged among the objects during the interaction.

The collaboration diagram can be a decomposition of a class, class diagram, or part of a class diagram.it can be the decomposition of a use case, use case diagram, or part of a use case diagram.

The collaboration diagram shows messages being sent between classes and object (instances). A diagram is created for each system operation that relates to the current development cycle (iteration).

METHODOLOGY

Pre-processing and training the model (CNN): The dataset is preprocessed such as Image reshaping, resizing and conversion to an array form. Similar processing is also done on the test image. A dataset consisting of about 4 different Paddy Leaves Disease is obtained, out of which any image can be used as a test image for the software.



The train dataset is used to train the model (CNN) so that it can identify the test image and the severity it has CNN has different layers that are Dense, Dropout, Activation, Flatten, Convolution2D, and MaxPooling2D. After the model is trained successfully, the software can identify the Paddy Leaves Disease Classification image contained in the dataset. After successful training and preprocessing, comparison of the test image and trained model takes place to predict the knee osteoarthritis

ARCHITECTURE OF CNN

CONVOLUTIONAL NEURAL NETWORK:

A Convolutional neural network (CNN) is one type of Artificial Neural Network. A Convolutional neural network (CNN) is a neural network that has one or more convolutional layers and are used mainly for image processing, classification, segmentation and also for other auto correlated data.

Models API:

There are three ways to create Keras models:

- The Sequential model, which is very straightforward (a simple list of layers), but is limited to single-input, single-output stacks of layers (as the name gives away).
- The Functional API, which is an easy-to-use, fully-featured API that supports arbitrary model architectures. For most people and most use cases, this is what you should be using. This is the Keras "industry strength" model.
- Model subclassing, where you implement everything from scratch on your own. Use this if you have complex, out-of-the-box research use cases

Types of Keras Models

Models in keras are available in two types:

- Keras Sequential Model
- Keras Functional API

1. Sequential Model in Keras

It allows us to create models layer by layer in sequential order. But it does not allow us to create models that have multiple inputs or outputs.

It is best for simple stack of layers which have 1 input tensor and 1 output tensor.

This model is not suited when any of the layer in the stack has multiple inputs or outputs. Even if we want non-linear topology, it is not suited.

2. Functional API in Keras

It provides more flexibility to define a model and add layers in keras. Functional API allows us to create models that have multiple input or output.

It also allows us to share these layers. In other words. we can make graphs of layers using Keras functional API.

As functional API is a data structure, it is easy to save it as a single file that helps in recreating the exact model without having the original code. Also its easy to model the graph here and access its nodes as well.

Model Subclassing in Keras

Sequential model does not allow you much flexibility to create your models. Functional API also only has a little of customization available for you. But you may create your own fully-customizable models in Keras. This is done by subclassing the Model class and implementing a call method.

`Input ()` is used to instantiate a Keras tensor.

A Keras tensor is a symbolic tensor-like object, which we augment with certain attributes that allow us to build a Keras model just by knowing the inputs and outputs of the model.

For instance, if `a`, `b` and `c` are Keras tensors, it becomes possible to

```
do: model = Model(input=[a, b], output=c)
```

kernels:

Each convolutional layer contains a series of filters known as convolutional kernels. The filter is a matrix of integers that are used on a subset of the input pixel values, the same size as the kernel. Each pixel is multiplied by the corresponding value in the kernel, then the result is summed up for a single value for simplicity representing a grid cell, like a pixel, in the output channel/feature map. These are linear transformations; each convolution is a type of affine function.

In computer vision the input is often a 3 channel RGB image. For simplicity, if we take a greyscale image that has one channel (a two dimensional matrix) and a 3x3 convolutional kernel (a two dimensional matrix). The kernel strides over the input matrix of numbers moving horizontally column by column, sliding/scanning over the first rows in the matrix containing the images pixel values. Then the kernel strides down vertically to subsequent rows. Note, the filter may stride over one or several pixels at a time, this is detailed further below.

In other non-vision applications, a one-dimensional convolution may slide vertically over an input matrix.

Padding:

To handle the edge pixels there are several approaches:

- Losing the edge pixels
- Padding with zero value pixels
- Reflection padding

Reflection padding is by far the best approach, where the number of pixels needed for the convolutional kernel to process the edge pixels are added onto the outside copying the pixels from the edge of the image. For a 3x3 kernel, one pixel needs to be added around the outside, for a 7x7 kernel then three pixels would be reflected around the outside. The pixels added around each side is the dimension, halved and rounded down.

Traditionally in many research papers, the edge pixels are just ignored, which loses a small proportion of the data and this gets increasing worse if there are many deep convolutional layers. For this reason, I could not find existing diagrams to easily convey some of the points here without being misleading and confusing stride 1 convolutions with stride 2 convolutions.

With padding, the output from a input of width w and height h would be width w and height h (the same as the input with a single input channel), assuming the kernel takes a stride of one pixel at a time.

Strides:

It is common to use a stride two convolution rather than a stride one convolution, where the convolutional kernel strides over 2 pixels at a time, for example our 3x3 kernel would start at position (1,1), then stride to (1,3), then to (1, 5) and so on, halving the size of the output channel/feature map, compared to the convolutional kernel taking strides of one.

With padding, the output from an input of width w , height h and depth 3 would be the ceiling of width $w/2$, height $h/2$ and depth 1, as the kernel outputs a single summed output from each stride.

Rectified Linear Unit (ReLU):

A Rectified Linear Unit is used as a non-linear activation function. A ReLU says if the value is less than zero, round it up to zero.

Normalisation:

Normalisation is the process of subtracting the mean and dividing by the standard deviation. It transforms the range of the data to be between -1 and 1 making the data use the same scale, sometimes called Min-Max scaling.

It is common to normalize the input features, standardising the data by removing the mean and scaling to unit variance. It is often important the input features are centred around zero and have variance in the same order. With some data, such as images the data is scaled so that it's range is between 0 and 1, most simply dividing the pixel values by 255.

Batch normalisation:

Batch normalisation has the benefits of helping to make a network output more stable predictions, reduce overfitting through regularisation and speeds up training by an order of magnitude.

Batch normalisation is the process of carrying normalisation within the scope activation layer of the current batch, subtracting the mean of the batch's activations and dividing by the standard deviation of the batches activations.

During training (i.e. when using `fit()` or when calling the layer/model with the argument `training=True`), the layer normalizes its output using the mean and standard deviation of the current batch of inputs. That is to say, for each channel being normalized, the layer returns $\gamma * (\text{batch} - \text{mean}(\text{batch})) / \sqrt{\text{var}(\text{batch}) + \text{epsilon}} + \text{beta}$, where:

- `epsilon` is small constant (configurable as part of the constructor arguments)
- `gamma` is a learned scaling factor (initialized as 1), which can be disabled by passing `scale=False` to the constructor.
- `beta` is a learned offset factor (initialized as 0), which can be disabled by passing `center=False` to the constructor.

During inference (i.e. when using `evaluate()` or `predict()` or when calling the layer/model with the argument `training=False` (which is the default), the layer normalizes its output using a moving average of the mean and standard deviation of the batches it has seen during training. That is to say, it returns $\gamma * (\text{batch} - \text{self.moving_mean}) / \sqrt{\text{self.moving_var} + \text{epsilon}} + \text{beta}$.

`self.moving_mean` and `self.moving_var` are non-trainable variables that are updated each time the layer is called in training mode, as such:

- $\text{moving_mean} = \text{moving_mean} * \text{momentum} + \text{mean}(\text{batch}) * (1 - \text{momentum})$
- $\text{moving_var} = \text{moving_var} * \text{momentum} + \text{var}(\text{batch}) * (1 - \text{momentum})$

As such, the layer will only normalize its inputs during inference *after having been trained on data that has similar statistics as the inference data*.

Arguments

- **axis**: Integer, the axis that should be normalized (typically the features axis). For instance, after a Conv2D layer with `data_format="channels_first"`, set `axis=1` in BatchNormalization.
- **momentum**: Momentum for the moving average.
- **epsilon**: Small float added to variance to avoid dividing by zero.
- **center**: If True, add offset of beta to normalized tensor. If False, beta is ignored.
- **scale**: If True, multiply by gamma. If False, gamma is not used. When the next layer is linear (also e.g. `nn.relu`), this can be disabled since the scaling will be done by the next layer.
- **beta_initializer**: Initializer for the beta weight.
- **gamma_initializer**: Initializer for the gamma weight.
- **moving_mean_initializer**: Initializer for the moving mean.
- **moving_variance_initializer**: Initializer for the moving variance.
- **beta_regularizer**: Optional regularizer for the beta weight.
- **gamma_regularizer**: Optional regularizer for the gamma weight.
- **beta_constraint**: Optional constraint for the beta weight.
- **gamma_constraint**: Optional constraint for the gamma weight.

Call arguments

- **inputs**: Input tensor (of any rank).
- **training**: Python boolean indicating whether the layer should behave in training mode or in inference mode.

14. CONCLUSION AND FUTURE ENHANCEMENT

14.1 CONCLUSION:

In this project, research to classify Paddy leaves images using deep learning techniques was developed. This is a complex problem that has already been approached several times with different techniques. While good results have been achieved using feature engineering, this project focused on feature learning, which is one of DL promises. While feature engineering is not necessary, image pre-processing boosts classification accuracy. Hence, it reduces noise on the input data. Nowadays, Paddy leaves disease classification includes the use of feature engineering. A solution totally based on feature learning does not seem close yet because of a major limitation. Thus, Paddy leaves disease classification could be achieved by means of deep learning techniques.

14.2 FUTURE WORK:

- To apply this method in cloud computing
- To apply this method in IOT system.

REFERENCES:

1. S. Aggarwal, A. Jindal, R. Chaudhary et al., “EnergyChain: enabling energy trading for smart homes using blockchains in smart grid ecosystem,” in *Proceedings of the 1st ACM MobiHoc Work. Netw. Cybersecurity Smart Cities, SmartCitiesSecurity 2018*, Los Angeles, USA, June 2018.
2. A. Darshan, N. Girdhar, R. Bhojwani et al., “Energy audit of a residential building to reduce energy cost and carbon footprint for sustainable development with renewable energy sources,” *Advances in Civil Engineering*, vol. 2022, Article ID 4400874, 10 pages, 2022.
3. L. Natrayan, P. Sakthi shunmuga sundaram, and J. Elumalai, “Analyzing the uterine physiological with MMG signals using SVM,” *International Journal of Pharmaceutical Research*, vol. 11, no. 2, pp. 165–170, 2019.
4. K. Ahmed, T. R. Shahidi, S. M. Irfanul Alam, and S. Momen, “Rice leaf disease detection using machine learning techniques,” in *Proceedings of the 2019 International Conference on. Sustainable Technologies for Industry 4.0*, Dhaka, Bangladesh, December 2019.
5. R. R. Atole and D. Park, “A multiclass deep convolutional neural network classifier for detection of common rice plant anomalies,” *International Journal of Advanced Computer Science and Applications*, vol. 9, pp. 67–70, 2018.
6. Anupama, “Deep learning with backtracking search optimization-based skin lesion diagnosis model,” *Computers, Materials & Continua*, vol. 70, no. 1, pp. 1297–1313, 2021.

15. APPENDIX

A. SOURCE CODE

Module – 4

DEPLOYMENT VIEWS

```
from django.shortcuts import render, redirect from
. models import UserPredictModel
from . forms import UserPredictForm, UserRegisterForm from
django.contrib.auth import authenticate, login,logout from
django.contrib import messages import numpy as np

from tensorflow import keras from
PIL import Image, ImageOps from .
import forms

def Landing_1(request):
    return render(request, '1_Landing.html')
def
Register_2(request):
    form = UserRegisterForm()
if request.method == 'POST':
    form = UserRegisterForm(request.POST)
if form.is_valid():
    form.save()
    user = form.cleaned_data.get('username')
messages.success(request, 'Account was successfully def
Login_3(request):
    if request.method == 'POST':
        username = request.POST.get('username')
        password = request.POST.get('password')

        user = authenticate(username=username, password=password)
```

```

        print('HIFORM')
form.save()          obj =
form.instance

        result1 = UserPredictModel.objects.latest('id')
models = keras.models.load_model('C:/Users/SPIRO-
PYTHON/Desktop/Projects/ITPDL10 -
RICE/DEPLOYMENT_2_DL/PROJECT/APP/RESNET.h5')
        data = np.ndarray(shape=(1, 224, 224, 3), dtype=np.float32)
image = Image.open("C:/Users/SPIROPYTHON/Desktop/Projects/ITPDL10 -
data[0] = normalized_image_array          classes =
['bacterial_leaf_blight','bacterial_leaf_streak','bacterial_panicle_
blight','blast','brown_spot','dead_heart','downy_mildew','hispa','no
rmal','tungro']
        prediction = models.predict(data)
idd = np.argmax(prediction)          a =
(classes[idd])          if a ==
'bacterial_leaf_blight':          a =
'bacterial_leaf_blight.'          elif a ==
'bacterial_leaf_streak':          a =
'bacterial_leaf_streak.'          elif a ==
'bacterial_panicle_blight':          a =
'bacterial_panicle_blight.'          elif a ==
'blast':          a = 'blast'          elif
a == 'brown_spot':          a =
'brown_spot.'
        elif a == 'dead_heart':
a = 'dead_heart'          elif a
== 'downy_mildew':          a
= 'downy_mildew'          elif a
== 'hispa':          a =
'hispa'          elif a ==

```

```

'normal':          a =
'normal'          elif a ==
'tungro':         a =
'tungro'          else:
                    a = 'WRONG INPUT'

                    data = UserPredictModel.objects.latest('id')
data.label = a      data.save()

                    return render(request,
'8_Deploy.html',{ 'form':form, 'obj':obj, 'predict':a})
else:
                    form = forms.UserPredictForm()
                    return render(request, '8_Deploy.html',{ 'form':form})

def Out_Database_9(request):
    models = UserPredictModel.objects.all()
    return render(request, '9_Out_Database.html', {'models':models})
def Logout(request):    logout(request)    return
redirect('Landing_1')

```

OUTPUT:

