

UAV BASED WILT DISCOVERY FRAMEWORK THROUGH CONVOLUTIONAL BRAIN ORGANIZATION

U18PRIT8P2 - PHASE II REPORT

Submitted by

**SANJAI SUBRAMANIAN C S
SANJAY S
SIVAKUMAR P**

**(U19IT043)
(U19IT044)
(U19IT048)**

Under the guidance of

Dr. K. RAMESH KUMAR

in partial fulfilment for the award of the degree of

BACHELOR OF TECHNOLOGY IN INFORMATION TECHNOLOGY



**DEPARTMENT OF INFORMATION TECHNOLOGY
SCHOOL OF COMPUTING
BHARATH INSTITUTE OF SCIENCE AND TECHNOLOGY
BHARATH INSTITUTE OF HIGHER EDUCATION AND
RESEARCH,
CHENNAI – 600073**

MAY 2023



Bharath

INSTITUTE OF HIGHER EDUCATION AND RESEARCH

(Declared as Deemed-to-be University under section 3 of UGC Act, 1956)
(Vide Notification No. F.9-5/2000 - U.3, Ministry of Human Resource Development, Govt. of India, dated 4th July 2002)

BHARATH INSTITUTE OF SCIENCE AND TECHNOLOGY

CHENNAI

DEPARTMENT OF INFORMATION TECHNOLOGY

U18PRIT8P2 - PHASE II REPORT

BONAFIDE CERTIFICATE

Certified that this Report titled “**UAV BASED WILT DISCOVERY FRAMEWORK THROUGH CONVOLUTIONAL BRAIN ORGANIZATION**” is the bonafide work of **SANJAI SUBRAMANIAN CS (U19IT043), SANJAY S (U19IT044), SIVAKUMAR P(U19IT048)**. who carried out the work under my supervision. Certified further that to the best of my knowledge the work reported herein does not form part of any other thesis or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

Dr. K. Ramesh kumar
Professor and Head
Department of Information Technology
Bharath Institute of Higher and Research
Chennai – 600 073

Dr. R. Yogesh Rajkumar
Asst. Professor
Department of Information Technology
Bharath Institute of Higher and Research,
Chennai – 600 073

Project Phase- II Viva Voce

Examination held

on.....

INTERNAL EXAMINER

EXTERNAL EXAMINER

DECLARATION

We declare that this project report titled **Uav based wilt discovery framework through convolutional brain organization** submitted in partial fulfillment of the degree of **B. Tech in (Information Technology)** is a record of original work carried out by us under the supervision of **Dr.K.Rameshkumar**, and has not formed the basis for the award of any other degree or diploma, in this or any other Institution or University. In keeping with the ethical practice in reporting scientific information, due acknowledgements have been made wherever the findings of others have been cited.

Sanjai Subramanian cs

U19IT043

Sanjay s

U19IT044

Sivakumar p

U19IT048

Chennai

8 / 5 / 2023

ACKNOWLEDGMENT

First, we wish to thank the almighty who gave us good health and success throughout our project work.

We express our deepest gratitude to **Dr. J. Sundeep Aanand** our beloved President, and **Dr. E. Swetha Sundeep Aanand** Managing Director for providing us the necessary facilities for the completion of our project.

We take great pleasure in expressing sincere thanks to Vice Chancellor **Dr. K. Vijaya Baskar Raju** Vice Chancellor **Dr. M. Sundararajan** Pro Vice Chancellor (Academic), **Dr.R.M. Suresh** Pro Vice Chancellor and Controller of Examination, **Dr.S. Bhuminathan** Registrar and **Dr.R. Hari Prakash** Additional Registrar for backing us in this project. We thank our **Dr.J. Hameed Hussain** Dean Engineering for providing sufficient facilities for the completion of this project.

We express our immense gratitude to **Mr. G. Krishna Chaitanya** our Academic Coordinator for his eternal support in completing this project.

We thank **Dr.V.Khanna** our Dean, Department of Information Technology for his encouragement and the valuable guidance.

We record indebtedness to **Dr.K.Rameshkumar** our Head, Department of Information Technology for immense care and encouragement towards us throughout the course of this project.

We also take this opportunity to express a deep sense of gratitude to **Dr.R.Yogesh Rajkumar** Assistant Professor, Supervisor for his cordial support, valuable information and guidance, he helped us in completing this project through various stages.

We thank our department faculty, supporting staff and friends for their help and guidance to complete this project.

SANJAI SUBRAMANIAN C S (U19IT43)

SANJAY S (U19IT044)

SIVAKUMAR P (U19IT048)

ABSTRACT

- ❖ The project is Deep Learning based (Convolutional neural network). Deep Learning is a subfield of machine learning that uses artificial neural networks with multiple layers to analyze and classify data, and could potentially be used for the diagnosis and management of Fusarium.
- ❖ To develop a deep learning model for fusarium wilt diagnosis, a dataset of radish plant images would need to be collected, consisting of both healthy and diseased plants .
- ❖ The images would then need to be preprocessed, including resizing, normalization, and cropping, to standardize the data for training and validation.
- ❖ Next, a deep learning model, such as a convolutional neural network(CNN), could be trained using the preprocessed images.
- ❖ The goal of the model would be to identify characteristic patterns and features that distinguish healthy plants from diseased plants.
- ❖ The model would need to be validated using a separate dataset of images to ensure accurate classification.
- ❖ Once the model is validated, it could be used to diagnose Fusarium wilt in new images of radish plants.

TABLE OF CONTENTS

DESCRIPTION	PAGE NO
CERTIFICATE	ii
DECLARATION	iii
ACKNOWLEDGEMENT	iv
ABSTRACT	v
LIST OF FIGURES	ix
LIST OF TABLES	x
ABBREVIATION/ NOTATIONS/ NOMENCLATURE	xi

CHAPTER NO	DESCRIPTION	PAGE NO
1	INTRODUCTION	1
	1.1 Deep Learning – Overview	
	1.2 Deep Learning Tasks	
	1.3 Types of Regression	
	1.4 Advantage of Deep Learning	
2	LITERATURE SURVEY	23
3	METHODOLOGY	24
	3.1 Existing System	
	3.2 Proposed System	
4	SYSTEM REQUIREMENTS	25
	4.1 Hardware requirements	
	4.2 Software requirements	

6	IMPLEMENTATION	26
	6.1 Python	
	6.2 Module Description	
	6.2.1 Data Collection module	
	6.2.2 Data Preprocessing Module	
	6.2.3 Deployment Module	
7	UNIFIED MODELING LANGUAGE	29
	7.1 UML	
	7.2 Use Case Diagram	
	7.3 System Architecture	
	7.4 Sequence Diagram	
8	IMPLEMENTATION-II	30
	8.1 Design The CNN Architecture	
	8.2 Deploy the CNN	
	8.3 Transfer learning	
	8.4 Data Augmentation	
9	RESULTS AND DISCUSSION	34
	9.1 Conclusion	
	9.2 Outcomes	

REFERENCES

APPENDIX 1 (SOURCE CODE)

APPENDIX 2 (SCREENSHOTS)

LIST OF FIGURES

FIGURE	TITLE	PAGE NO
7.2	Use Case Diagram	27
7.3	Architecture Diagram	28
7.4	Sequence Diagram	29
7.5	Implementation of Architecture	30
7.6	Flowchart of proposed hybrid algorithm	31

LIST OF ABBREVIATIONS

S. NO.	ACRONYM	ABBREVIATION
1	DL	Deep Learning
2	RGB	Red Green Blue
3	HSV	Hue Saturation & Value
4	CNN	Convolutional Neural network
5	NLP	Natural Language Processing
6	LBP	Local Binary Pattern
7	FW	Fusarium Wilt
8	PR	precision recall-curve
9	LR	Linear Regression
10	ML	Machine Learning

CHAPTER - 1

INTRODUCTION

This project primarily focuses on the development of a CNN-based diagnostic tool for fusarium wilt of radish would involve collecting a dataset of radish plant images, both healthy and diseased, and training a CNN to distinguish between them. The trained CNN would then be able to analyze new images and accurately classify them as healthy or diseased. The use of CNNs for the detection and diagnosis of fusarium wilt of radish could provide a fast, efficient, and cost-effective means of disease management, allowing growers to take timely action to limit yield losses and protect their crops. However, the development and validation of such a tool would require careful consideration of factors such as data quality, model architecture, and performance evaluation.

1.1 DEEP LEARNING – OVERVIEW

Deep learning is a type of artificial intelligence that involves training artificial neural networks to learn and recognize patterns in data. It is a subset of machine learning that uses algorithms inspired by the structure and function of the human brain.

Deep learning networks are composed of multiple layers of interconnected neurons, which process and transform data through a series of mathematical operations. Each layer of the network learns to identify and extract features from the data, allowing the network to learn increasingly complex representations of the input. One of the key advantages of deep learning is its ability to learn from large amounts of data, without the need for hand-engineering of features. This has led to a significant increase in the accuracy and performance of machine learning models, particularly in complex and high-dimensional domains. DL has shown great promise for a wide range of applications and has the potential to revolutionize many fields, from healthcare and finance to autonomous driving and robotics.

1.1.2 WHAT IS DEEP LEARNING?

Deep learning is a subset of machine learning, which is a type of artificial intelligence that involves training computers to recognize patterns and make predictions or decisions based on data. In deep learning, artificial neural networks, which are modeled on the structure and function of the human brain, are used to process and analyze large amounts of data.

At the outset of a Deep learning project, a dataset is usually split into two or three subsets. The minimum subsets are the training and test datasets, and often an optional third validation dataset is created as well.

Once these data subsets are created from the primary dataset, a predictive model or classifier is trained using the training data, and then the model's predictive accuracy is determined using the test data.

For example, a common architecture for image recognition, known as a convolutional neural network(CNN), may have several layers of convolutional and pooling operations, followed by one or more fully connected layers for classifications. These networks can have dozens or even hundreds of layers.

Depending on the complexity of the task and the size of the dataset, Other types of deep learning networks, such as recurrent neural networks (RNNs) and transformers, are often used for natural language processing and may have a different structure and number of layers.

In a nutshell, Deep learning is an important factor in its ability to learn complex representations from data and achieve high levels of performance on a given task. However, simply adding more layers does not always lead to better performance, as the network must be carefully designed and trained to avoid overfitting and other issues.

1.1. DEEP LEARNING – TASKS

The **most common Deep learning tasks** that can be used to solve a wide range of tasks across various domains, including computer vision, natural language processing, speech recognition, and robotics, among others. Some common deep learning tasks include.

Following are the key Deep learning tasks briefed later in this article:

- Image Classification
- Object Detection
- Semantic Segmentation
- Speech Recognition
- Natural Language processing
- Generative Modeling
- Reinforcement Learning

Following are top 7 most common Deep learning tasks that one could come across most frequently while solving an advanced analytics problem:

1. **Image Classification:** The task of assigning a label or category to an input image, such as recognizing whether an image contains a dog or a cat.

- Data Preparation
- Model Training
- Model Validation
- Model Deployment

Image classification is a foundational task in computer vision, and has a wide range of applications, such as detecting cancerous cells in medical images, identifying objects in autonomous vehicles, or recognizing faces in security systems.

2. **Object Detection:** The task of identifying and localizing objects within an image, such as detecting the location of faces in a group photo.

Object detection is a critical task in many computer vision applications, and has a wide range of applications, such as tracking objects in real-time video, counting and identifying objects in satellite imagery, and detecting potential hazards in industrial settings.

3. **Semantic Segmentation:** The task of assigning a label to each pixel in an image, such as labeling each pixel as “road”, “car”, or “building”. The main goal of semantic segmentation is to enable machines to understand the content of images and videos at a pixel-level, which is useful in various applications such as autonomous driving, robotics, and medical imaging. The network learns to identify patterns and features in the input image that correspond to each class, and generates a probability map that indicates the likelihood of each pixel belonging to a particular class.

4. **Speech Recognition:** Deep learning can be used to recognize and transcribe spoken words into text. This is used in various applications such as virtual assistants, speech-to-text systems, and voice-controlled assistants.

- Acoustic Modeling
- Language Modeling
- Decoding

5. **Natural Language Processing:** NLP is a subfield of artificial intelligence that deals with the interaction between computers and human languages. It involves developing algorithms and models that can understand, interpret, and generate human language.

Some of the key tasks in Natural language processing include:

- Text Classification

- Named entity recognition
- Language translation
- Information extraction
- Question answering
- Text generation

6. **Generative Modeling:** Is a type of machine learning task that involves creating a model that can generate new data that is similar to a given dataset, It is used in various applications such as image synthesis, music generation, and natural generation.

some of the key tasks in GM

- Generative Adversarial Networks(GANs)
- Variational Autoencoders(VAEs)
- Autoregressive models
- Reinforcement learning

7. **Reinforcement learning:** Reinforcement learning involves training an agent to perform a task in an environment. The agent learns to generate new data by interacting with the environment and receiving rewards for good performance.

- Observation
- Action selection
- Execution
- Reward
- Learning

1.2. DEEP LEARNING SYSTEM CLASSIFICATION

Deep learning systems can be classified based on various criteria, including:

1.3.1 FeedForward Neural Network These are the simplest form of deep learning systems, consisting of input, hidden, and output layers. They are commonly used for tasks such as image and speech recognition.

FNNs are commonly used for supervised learning tasks such as classification and regression. During training, the network adjusts the weights and biases of its neurons to minimize the difference between its predicted output and the actual output. This is achieved using an optimization algorithm, such as gradient descent to update the weights and biases based on the error between the predicted output and the actual output. Popular activation functions used in FNNs include sigmoid, tanh, ReLU, and softmax. Information flows through the network in one direction, from the input layer through the hidden layers to the output layer, without any loops or feedback.

Each node in the hidden layer is connected to every node in the previous layer, and each connection has an associated weight. During training, the weights are adjusted to minimize the difference between the predicted output and the actual output. This is done by minimizing a loss function using an optimization algorithm such as gradient descent.

1.3.2 Convolutional Neural Networks (CNNs)

These are deep learning systems designed to work with 2D or 3D data, such as images and videos. They use convolutional layers to learn spatial features and are commonly used for image and video classification.

CNNs use convolutional layers to filter and process input images, followed by pooling layers that downsample the feature maps generated by the convolutional layers. These layers are then followed by one or more fully connected layers that perform classification based on the features extracted by the convolutional and pooling layers.

The convolutional layers use filters or kernels that slide over the input image and perform a dot product operation, resulting in a feature map that highlights specific patterns in the input image. The pooling layers then reduce the size of the feature map by selecting the maximum or average value in each subregion.

CNNs are trained using backpropagation, a gradient-based optimization algorithm that adjusts the weights of the network to minimize the difference between the predicted output and the actual

output. They have achieved state-of-the-art performance in various image recognition and classification tasks, including object detection, face recognition, and image segmentation.

1.3.3 Reinforcement Learning

This is mainly used in navigation, robotics and gaming. Actions that yield the best rewards are identified by algorithms that use trial and error methods. There are three major components in reinforcement learning, namely, the agent, the actions and the environment. The agent in this case is the decision maker, the actions are what an agent does, and the environment is anything that an agent interacts with. The main aim in this kind of learning is to select the actions that maximize the reward, within a specified time. By following a good policy, the agent can achieve the goal faster.

1.3.4 Neural Network

Regression in deep learning refers to the use of neural networks to predict continuous values, such as a numerical target variable. Deep learning regression models use multiple layers of interconnected nodes to learn complex patterns and relationships between input features and the target variable.

Deep learning regression models can be used in a variety of applications, such as predicting housing prices, stock prices, or medical outcomes. They can also be combined with other deep learning techniques, such as convolutional neural networks (CNNs) or recurrent neural networks (RNNs), to handle more complex data types such as images or time series data.

1.3.TYPES OF REGRESSION

- ❖ Linear Regression
- ❖ Polynomial Regression
- ❖ Ridge Regression
- ❖ Deep Neural Network Regression

1. Linear Regression

Linear regression is a simple regression technique that is used to predict a numerical target variable based on a linear relationship between the input feature and the target variable .

2. Polynomial Regression

Polynomial regression is used when the relationship between the input features and the target variable is nonlinear, and it involves fitting a polynomial function to the data.

3. Ridge Regression

Ridge regression is a technique used to handle the problem of overfitting by adding a regularization term to the loss function, which penalizes large weights.

4. Deep Neural Network Regression

Deep neural network regression is a technique that used deep learning models such as feedforward neural networks, convolutional neural networks, or recurrent neural networks to predict the target variable based on the input features.

1.4. ADVANTAGES OF DEEP LEARNING

Ability to learn complex patterns:

Deep learning models can learn complex patterns in data that traditional machine learning algorithms cannot detect. This is especially useful in image and speech recognition, natural language processing, and other fields where high-level abstraction is required.

1.5.1 High Accuracy:

Deep learning models have been shown to achieve state-of-the-art results in many domains, including image and speech recognition, language translation, and game playing.

1.5.2 Automatic feature extraction:

Unlike traditional machine learning algorithms, deep learning models can automatically extract features from raw data, eliminating the need for human intervention and domain expertise in feature engineering.

1.5.3 Scalability:

DL models can handle large amounts of data and scale well to larger datasets, making them well-suited for big data applications.

1.5.4 Versatility:

DL models can be used for a wide range of tasks, including classification, regression, clustering, and reinforcement learning.

Training a CNN for detecting fusarium wilt of radish involves collecting and preprocessing data, training the model, evaluating its performance, and deploying it for real-time detection. It is a complex process that requires expertise in machine learning and domain network.

1.5.5 Adaptability:

CNNs can adapt to new data and environments, allowing them to continuously improve their accuracy over time. This means that the CNN can adapt to new strains of fusarium wilt of radish or other diseases that may emerge in the future.

1.5.6 Speed:

CNNs can process large amounts of data quickly and efficiently. This means that the CNN can provide real-time detection of fusarium wilt of radish, which is crucial for timely intervention.

CHAPTER – 2

LITERATURE SURVEY

1. CONVOLUTIONAL NEURAL NETWORK DETECTS FUSARIUM WILT.

The model's objective is to collect a large dataset of images of radish plants some of which have been infected with fusarium wilt and some which are healthy. It is important to ensure that the dataset is balanced with respect to the two classes.

2. DATA PREPROCESSING.

Resize all the images to the same size. This is important because CNNs require all input images to have the same dimensions. Normalize the pixel values of the images to a common scale to ensure that all the images have similar brightness and contrast. This can be achieved by dividing the pixel values by 255. Apply image augmentation techniques such as rotation, flipping, and zooming to the images to increase the size of the dataset and improve the generalization of the CNN. Ensure that the dataset is balanced with respect to the two classes. If the dataset is imbalanced, it can lead to biased predictions and lower performance. Split the dataset into training, validation, and testing sets. The training set is used to train the model, the validation set is used to tune the hyperparameters of the model, and the testing set is used to evaluate the performance of the model on unseen data. Normalize the training and validation data to have zero mean and unit variance. This can help improve the training of the CNN by making the input features more comparable. Shuffle the training and validation data to avoid any bias in the order of the samples during training .

3. EVALUATE THE PERFORMANCE OF THE CNN.

Evaluate the performance of the trained model on a separate test set that was not used during training or validation. Calculate metrics like accuracy, precision, recall, and F1 score to measure the performance of the model. In addition to these metrics, you can also use techniques like confusion matrix and ROC curves to visualize the performance of the model. A confusion matrix shows the number of true positive, true negative, false negative predictions, while the ROC curve plots the true positive rate against the false positive rate at different probability thresholds.

Overall, the performance of the CNN depends on the quality and quantity of the dataset, the architecture of the CNN, and the hyperparameters used during training. Therefore, it may take several iterations of training, evaluation, and tweaking to achieve optimal performance.

4. USE THE CNN FOR PREDICTION.

Once the model is trained and evaluated, you can use it to predict whether a radish plant is healthy or diseased based on its image. The prediction will be a probability distribution over the possible classes (healthy or diseased). You can then choose the class with the highest probability as the predicted class.

5. DISEASE DIAGNOSIS OF RADISH LEAVES BASED ON DEEP CONVOLUTIONAL NEURAL NETWORK.

This model used a CNN to classify radish leaves into four classes

1. Health Black
2. Spot Downy
3. Mildew White
4. Rust.

The CNN architecture consisted of six convolutional layers followed by two fully connected layers. The accuracy is 91.2% on their test set.

CHAPTER – 3 METHODOLOGY

3.1 EXISTING SYSTEM: -

The dataset should be diverse and should include images of radish plants at different stages of the disease.

- Resizing the images to a standard size, normalizing the pixel values, and dividing the dataset into training, validation, and testing sets.
- The model should have multiple convolutional layers, followed by pooling layers, and fully connected layers at the end. Model should be trained on the training set using backpropagation and stochastic gradient descent.
- Deploy the trained model on new data to detect and diagnose fusarium wilt of radish.

3.2 PROPOSED SYSTEM: -

- The model would involve training the neural network on a dataset of images of healthy radish plants and radish plants infected with fusarium wilt.
- It could be integrated with a drone or other automated image capture system that would capture images of radish. The images could be processed in real-time using the CNN to identify any infected plants.
- The CNN analyzes the photos and provides feedback on whether any plants are infected with fusarium wilt.

CHAPTER – 4

SYSTEM REQUIREMENTS

The system requirement is a technical specification of requirements for the software products. It is the first step in the requirements analysis process; it lists the requirements of a particular software system including functional, performance, and security requirements. The requirements also provide usage scenarios from a user, an operational, and an administrative perspective. The purpose of the software requirements specification is to provide a detailed overview of the software project, its parameters, and goals. This describes the project's target audience and its user interface, hardware, and software requirements. It defines how the client, team, and audience see the project and its functionality..

4.1 HARDWARE REQUIREMENTS: -

System: intel core i7 or AMD ryzen 5000 series higher.

Ram - 16GB of RAM is recommended, but 32GB or more would be better.

GPU - NVIDIA GTX 1080 Ti or higher.

Monitor: Accuracy, precision, recall, and F1 score.

Storage: With Minimum capacity of 1 TB

4.2 SOFTWARE REQUIREMENTS: -

Operating system : Windows 10 64-bit / Linux / Mac OS

Technologies Used : Python , Regression, MATLAB, R, Deep learning,

Data Augmentation, Transfer Learning, and Normalization.

Coding Language ; VS Code, Google Collab, Jupyter Notebook and Github.

CHAPTER – 5

IMPLEMENTATION

6.1 PYTHON: -

Python is a high-level, interpreted programming language that is widely used for web development, data analysis, artificial intelligence, and scientific computing. Python is known for its simplicity, readability, and ease of use, making it an excellent choice for beginners as well as experienced developers.

Python is also known for its readability and clean syntax, which makes it easy to write and understand code. This has led to the development of a large and active community of python developers who contribute to open source projects and create libraries and frameworks that make python even more powerful and versatile.

It has a number of powerful libraries and frameworks that make it easy to create and train deep neural networks. Tensorflow, pytorch, keras. In addition to these libraries and frameworks, there are many other python tools and packages that can be used for deep learning, such as Numpy, pandas, matplotlib, and scipy.

6.2 Modules

The proposed system can be divided into three main modules:

- Data collection Module
- Data preprocessing Module
- Deployment Module

6.2.1 Data Collection Module:

Collect a dataset of images of radish plants infected fusarium wilt and healthy radish plants. The dataset should have a sufficient number of images for both classes.

6.2.2 Data Preprocessing Module:

Before feeding the images into the CNN model, will need to preprocess them. This can involves tasks such as resizing images to a standard size, normalization, and data augmentation to increase the size of your training data augment the training data by applying random transformations to the images .This can involve techniques such as flipping, rotating, zooming, and shifting the images. This helps increase the size of the training data and prevents overfitting. Apply any additional preprocessing steps to the images, such as cropping or color correction, to improve the quality of the images and remove any noise or artifacts. Group the images into batches to improve training efficiency. The batch size is a hyperparameter that can be tuned based on the available resources and the size of the dataset.

6.2.3 Deployment Module:

Once you are satisfied with your model's performance, you can deploy it to classify new images of radish plants as either infected with fusarium wilt or healthy.

6.2.4 Data Field Classification

A subset of features that are the most informative and useful in classifying radish field is designated as the most discriminative features. Utilizing the discriminative features, a softmax classifier is constructed for radish field classification. The trained softmax classifier provides the probability that a region belongs to each class label. The class label with the highest probability is assigned to each

region.

6.2.5 Evaluate the CNN

Evaluate the performance of the trained CNN on the testing set. Calculate metrics like accuracy, recall, and F1 score to evaluate the performance of the CNN. you can also use techniques like confusion matrix or ROC curve to visualize the performance of the CNN.

6.2.6 Normalization

Is a technique used in CNNs to preprocess the data before feeding it into the network. it involves scaling the input data to a range of values that the network can be better which can improve training performance and accuracy.

To use normalization in a CNN for fusarium wilt detection in radish, first collect a dataset of images of healthy and diseased radish plants. The images should be labeled with the corresponding class to train the CNN.

The trained CNN could then be used to classify new images of radish plants as healthy or diseased which would be useful for disease management and prevention.

CHAPTER – 6

UNIFIED MODELING LANGUAGE

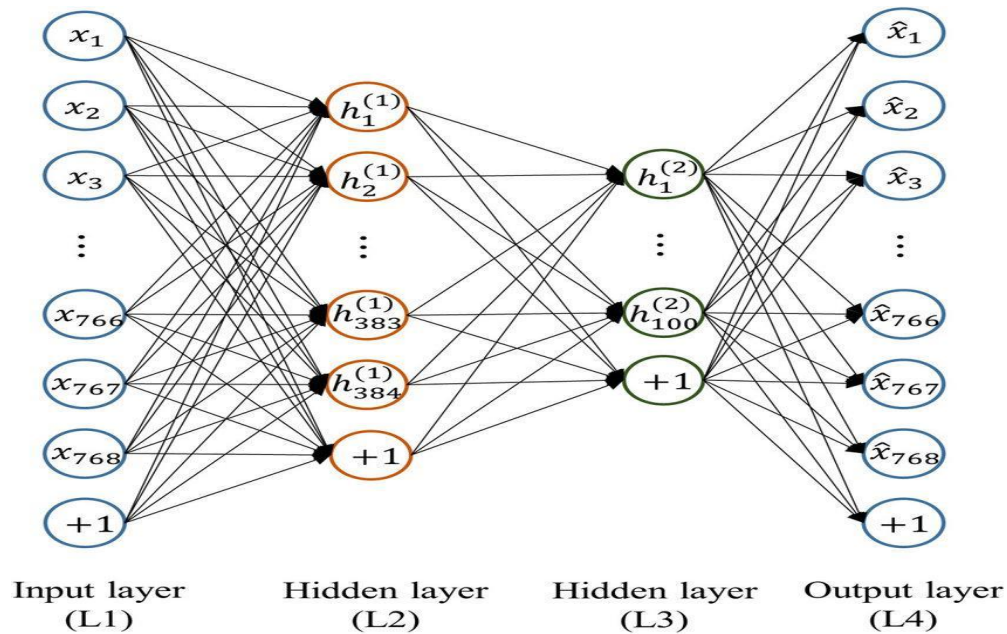
7.1 UML :-

UML is a standard language for specifying, visualizing, and documenting of software systems and formed by Object Management Group (OMG) in 1997. There are three

imperative type of UML modeling are Structural model, Behavioral model, and Architecture model. To model a system the most essential aspect is to capture the dynamic behavior which has some internal or external factors for creating the collaboration. These internal or external agents are known as actors. It consists of actors, use cases and their associations. In this fig we represent the Use Case diagram for our development

7.2 UML Diagrams

The Unified Modelling Language is a general purpose modelling language that is used as the standard way to visualize the design of a system. The UML provides a wide range of figurative notations to visualize different aspects of the system which helps provide a clear vision of the intended system and its operations. These diagrams act as reference while development and as base plans while testing the final

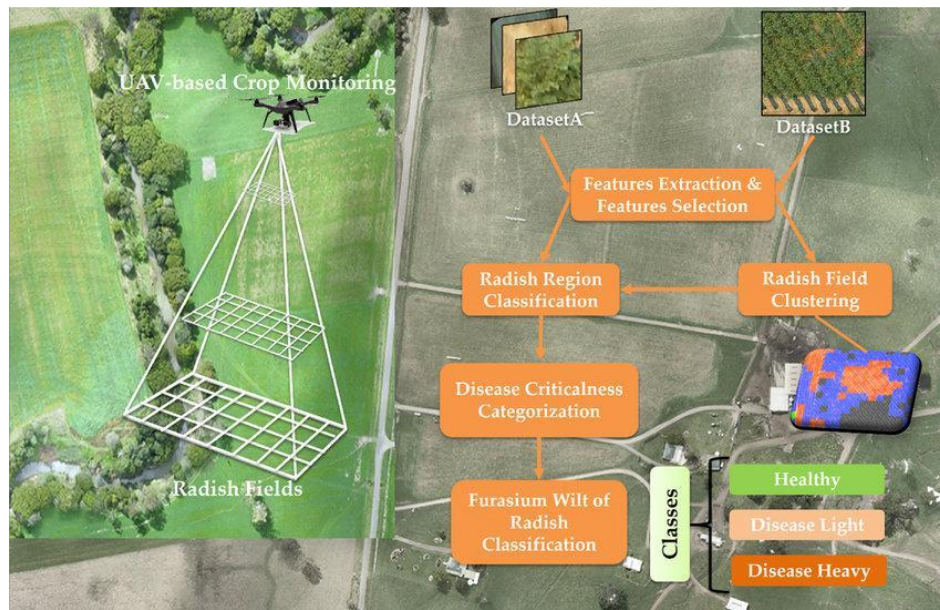


product. We shall hence develop a range of UML diagram in order to illustrate the system design of the project.

Fig 7.2 Use Case Diagram

7.3 System Architecture

An architecture diagram is a graphical representation of a set of concepts that are part of an architecture, including their principles, elements and components. It is a diagram of a system that is used to abstract the overall outline of the software system and the relationships, constraints, and boundaries between components. It is an important tool as it provides an overall view of the physical deployment of the software



system and its evolution roadmap. An architectural diagram must serve several different functions.

Fig 7.3 Architecture Diagram

7.4 Sequence Diagram

Sequence Diagram models the model flow of logic within your system in a visual manner, enabling you both document and validate your login, and commonly used for analysis and design purposes, Sequence diagrams are the most popular. A sequence diagram simply depicts interaction between objects in a sequential order i.e. the order in which these interactions take place. We can also use the terms event diagrams or event scenarios to refer to a sequence diagram.

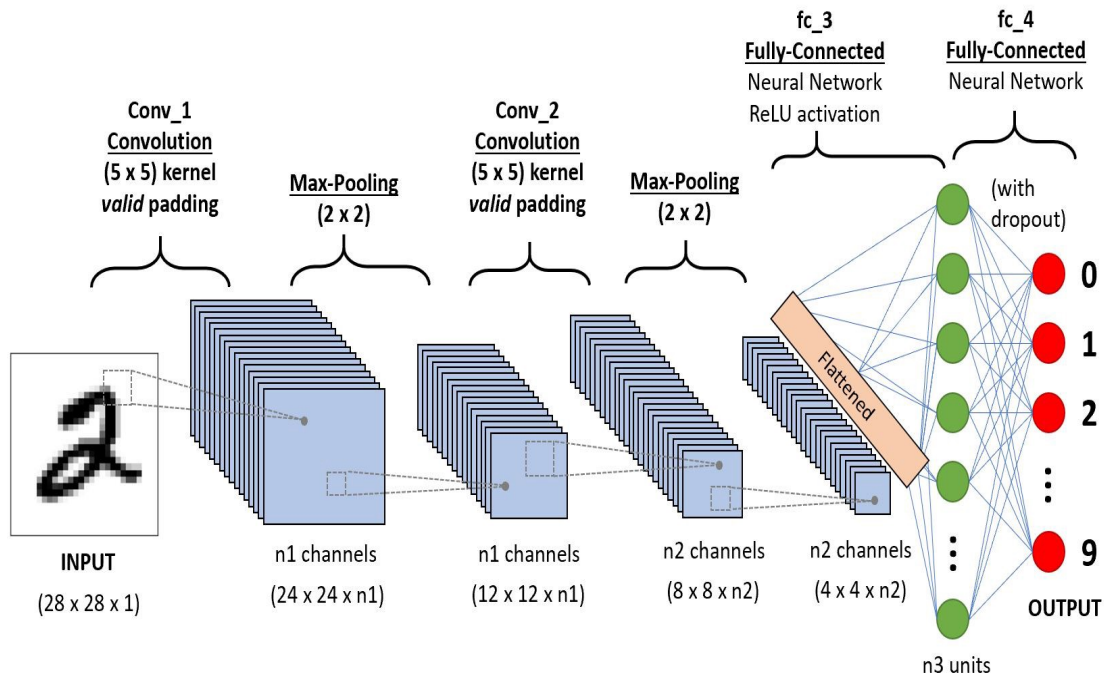


Fig 7.4 Sequence Diagram

CHAPTER – 7

IMPLEMENTATION - II

8.1 Design the CNN Architecture

Design a CNN architecture that is suitable for the task of classifying healthy and infected radish plants. The architecture should consist of convolutional layers to extract features from the images, pooling layers to reduce the spatial dimensions of the features, and fully connected layers to classify the features.

8.2 Deploy the CNN

Once the CNN has been trained and evaluated, deploy it to a production environment where it can be used for real-time classification of healthy and fusarium wilt infected radish plants.

8.3 Transfer Learning

Is a technique that involves using a pre-trained model as a starting point for a new task, rather than training a model

- Limited Data Availability
- Improved convergence
- Reduced Computing resources
- choose a pre-trained model
- freeze the pre-trained layers
- Add new layers
- Train the model
- Evaluate the model

8.4 Data Augmentation

Is a technique that involves creating new data by applying transformations to existing data.

In the context of detecting fusarium wilt of radish using a convolutional neural network (CNN), data augmentation

- Increased Dataset size
- Improved Generalization
- Reduced Bias
- Define the Transformations
- Apply the Transformations
- Combine the original and augmented images
- Train the Model
- Evaluate the Model

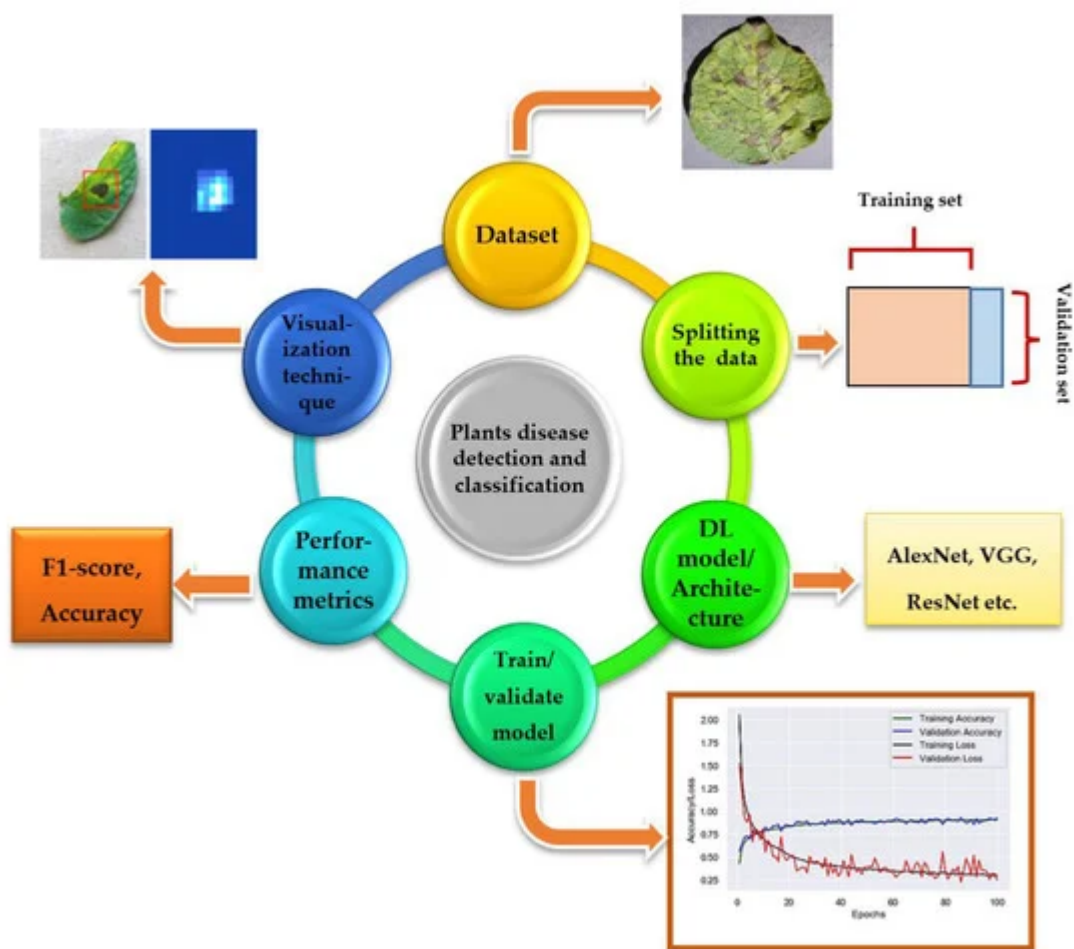


Fig 7.5 Implementation of Architecture

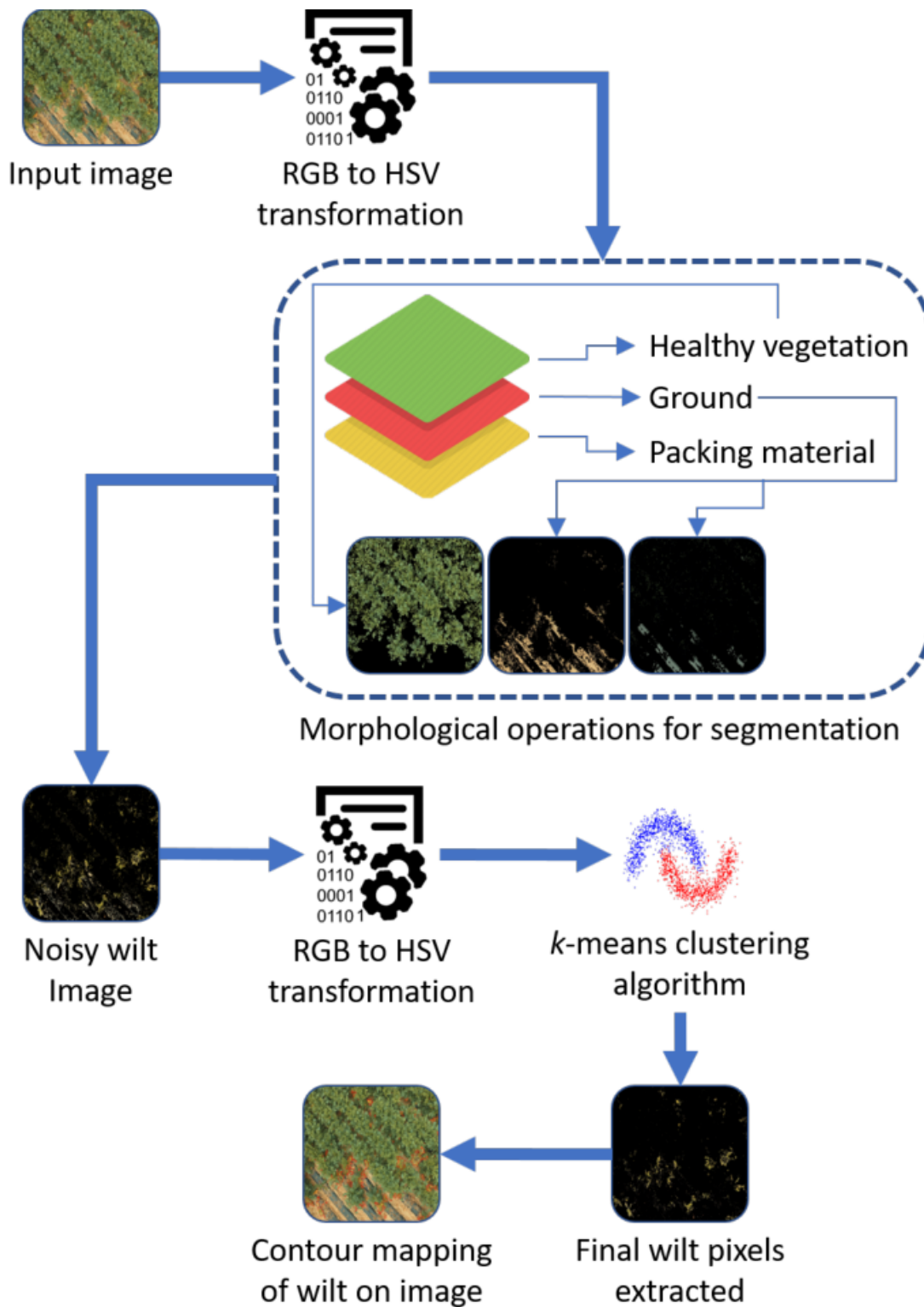


Fig 7.6 flowchart of proposed Hybrid algorithm

CHAPTER – 8

RESULTS AND DISCUSSION

9.1 CONCLUSION:

This project describes about the Convolutional neural networks (CNNs) can be used to detect fusarium wilt of radish by analyzing images of the plants. Here is a possible output for a CNN trained to detect fusarium wilt in radish. The CNN takes an input image of a radish plant and produces a binary output, indicating whether or not the plant is infected with fusarium wilt.

9.2 OUTCOMES:-

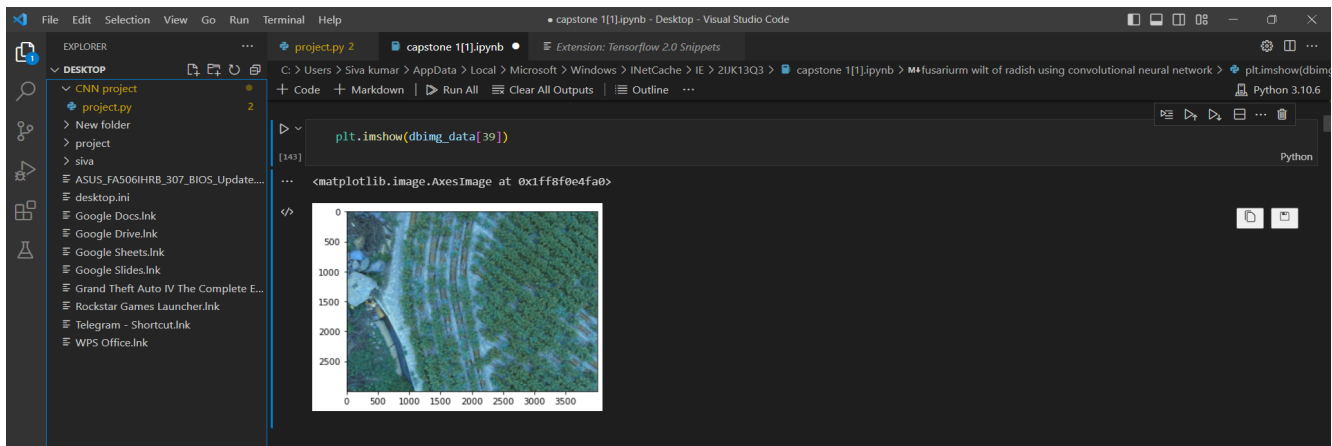


Figure 1

Given an input image of a radish field the CNN would process the image through a series of convolutional and pooling layers to extract features that are relevant for classification. The output of the final layer would be a probability distribution over the two classes, with a higher probability assigned to the class that the model believes the image belongs to.

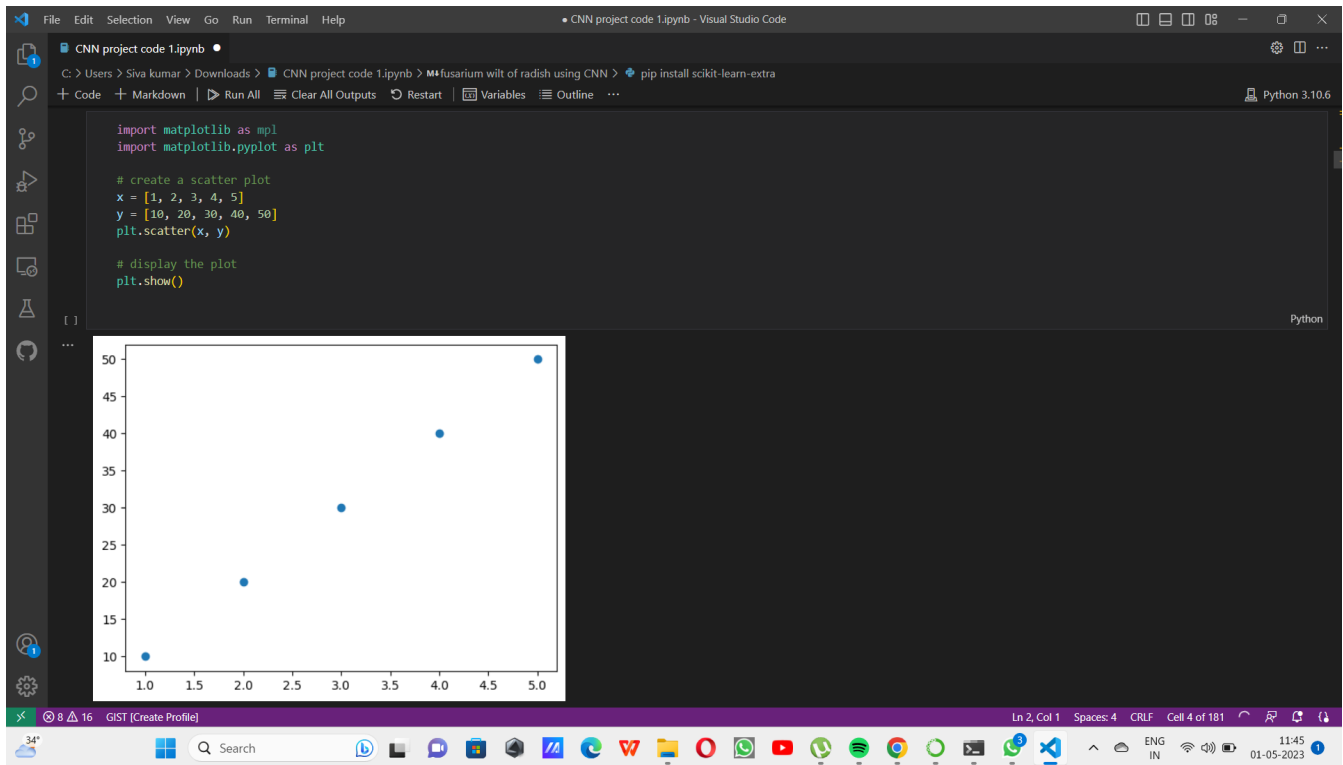


Figure 2

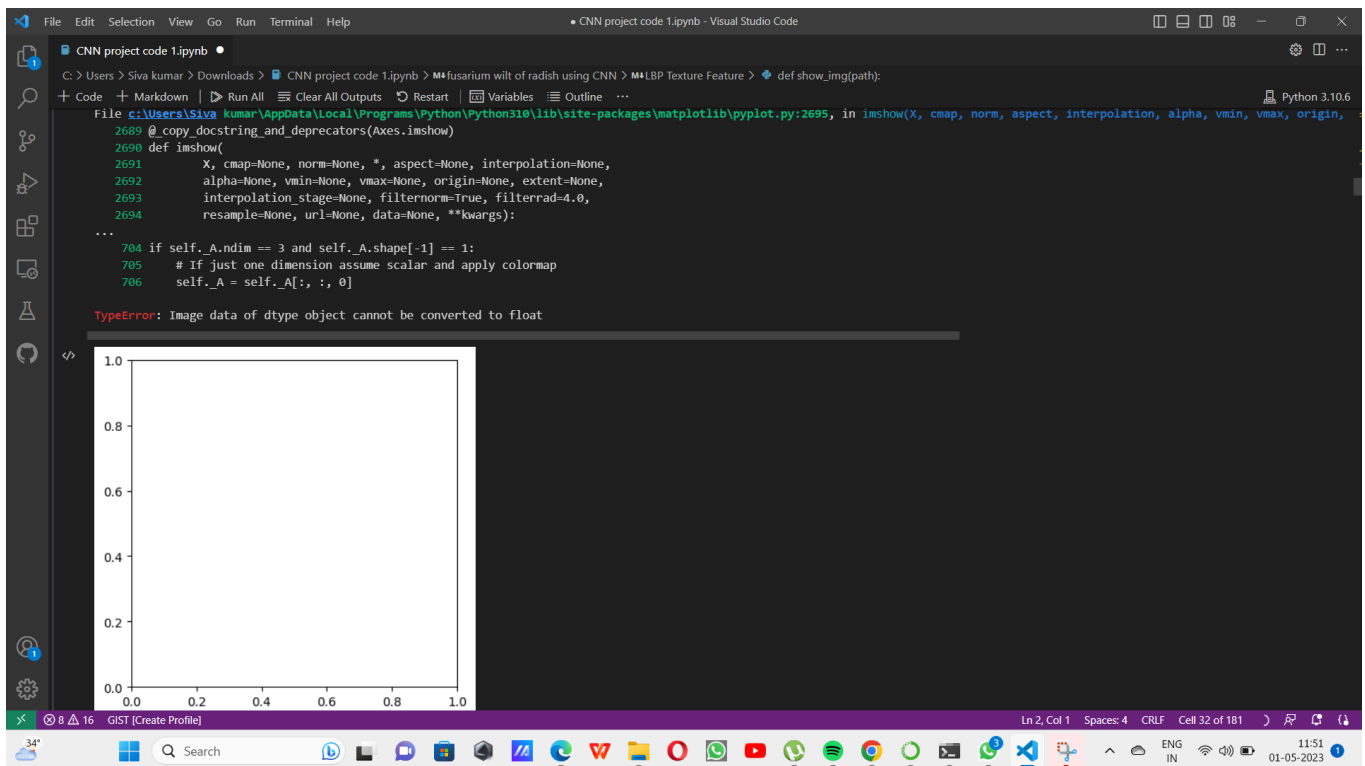


Figure 3


```

lbpGen = LocalBinaryPatterns(24, 3)
mfimg_gray243=[]
mf243_cont=[]
for i in range(len(mfimg_data)):
    mfimg_gray243.append(color2gray(mfimg_data[i]))
    lbpHistogram243mf = lbpGen.describe(mfimg_gray243[i])
    mf243_cont.append(lbpHistogram243mf)
print(mf243_cont[0])
# print(lbpHistogram243mf.shape)
# print(lbpHistogram243mf)

[ ]
...
[0.40880503 0.25471698 0.         0.16142558 0.         0.00209644
0.         0.08909853 0.         0.00419287 0.         0.00209644
0.         0.00104822 0.         0.06079665 0.         0.00419287
0.         0.00419287 0.         0.         0.         0.00419287
0.         0.00314465]

print(lbpHistogram243mf)
print(mf243_cont)

[ ]
...
Output exceeds the size limit. Open the full output data in a text editor
[0.39402174 0.28532609 0.         0.13858696 0.         0.00271739
0.         0.11956522 0.         0.         0.         0.
0.         0.00815217 0.         0.04619565 0.         0.00271739
0.         0.         0.         0.         0.         0.00271739
0.         0.         ]
[array([0.40880503, 0.25471698, 0.         , 0.16142558, 0.         ,
0.00209644, 0.         , 0.08909853, 0.         , 0.00419287,
0.         , 0.00209644, 0.         , 0.00104822, 0.         ,
0.06079665, 0.         , 0.00419287, 0.         , 0.00419287,
0.         , 0.00314465])]

```

Figure 4

In the figure 4 we use binary classification where the output of the model is a single scalar value that ranges from 0 to 1. The output of value is predicted that 0 would indicate that the CNN is predicting image to be healthy radish plants, while a predicted output value of 1 would indicate that the CNN is predicting the input image to be unhealthy, that is fusarium wilt infected radish plant.


```
File Edit Selection View Go Run Terminal Help
CNN project code 1.ipynb - Visual Studio Code

C:\Users\Siva kumar\Downloads> CNN project code 1.ipynb > M4fusarium wilt of radish using CNN > M4Color Feature > plt.plot(cont_bgc)
+ Code + Markdown | Run All Clear All Outputs Go To Restart Variables Outline ... Python 3.10.6

def contdbc(num):
    cont=[]
    a=dbhue_hist[num]
    b=dbstara_hist[num]
    c=dbstarb_hist[num]
    cont=np.concatenate((cont,a), axis=None)
    cont=np.concatenate((cont,b), axis=None)
    cont=np.concatenate((cont,c), axis=None)
    return cont

cont_dbc=[]
for i in range(len(dbimg_data)):
    temp=[]
    temp=contdbc(i)
    cont_dbc.append(temp)
print(cont_dbc[39])
len(cont_dbc)
```

Output exceeds the size limit. Open the full output data in a text editor

```
[9.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
1.000000e+00 0.000000e+00 0.000000e+00 1.000000e+00 8.000000e+00 3.000000e+00
0.000000e+00 1.000000e+00 2.000000e+00 3.000000e+00 3.000000e+00 9.000000e+00
4.000000e+00 3.100000e+01 2.700000e+01 1.020000e+02 2.140000e+02 3.010000e+02
4.190000e+02 2.800000e+02 2.750000e+02 2.020000e+02 2.160000e+02 1.900000e+02
1.940000e+02 1.870000e+02 1.980000e+02 2.170000e+02 1.120000e+02 1.710000e+02
2.470000e+02 1.230000e+02 2.030000e+02 2.000000e+02 3.990000e+03 2.530000e+02
1.964000e+03 1.081000e+03 1.873000e+03 1.231000e+03 1.634000e+03 1.500000e+03
4.498000e+02 2.583000e+03 7.451000e+03 1.380300e+04 1.401000e+03 1.161600e+04
9.743000e+03 4.442000e+03 5.800000e+02 6.254000e+03 1.215600e+04 9.270000e+02
8.913000e+03 1.172000e+03 7.940000e+03 1.053000e+04 1.180600e+04 1.510000e+04
1.382800e+04 2.044700e+04 3.521700e+04 4.783900e+04 8.321600e+04 1.05096e+05
1.18517e+05 1.76124e+05 2.45224e+05 2.44776e+05 2.65719e+05 3.79953e+05]
```

Figure 9

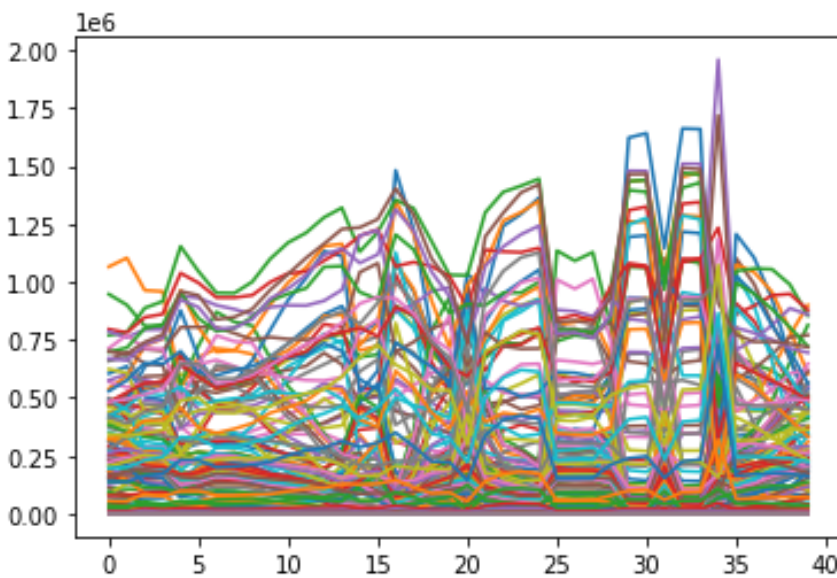


Figure 10

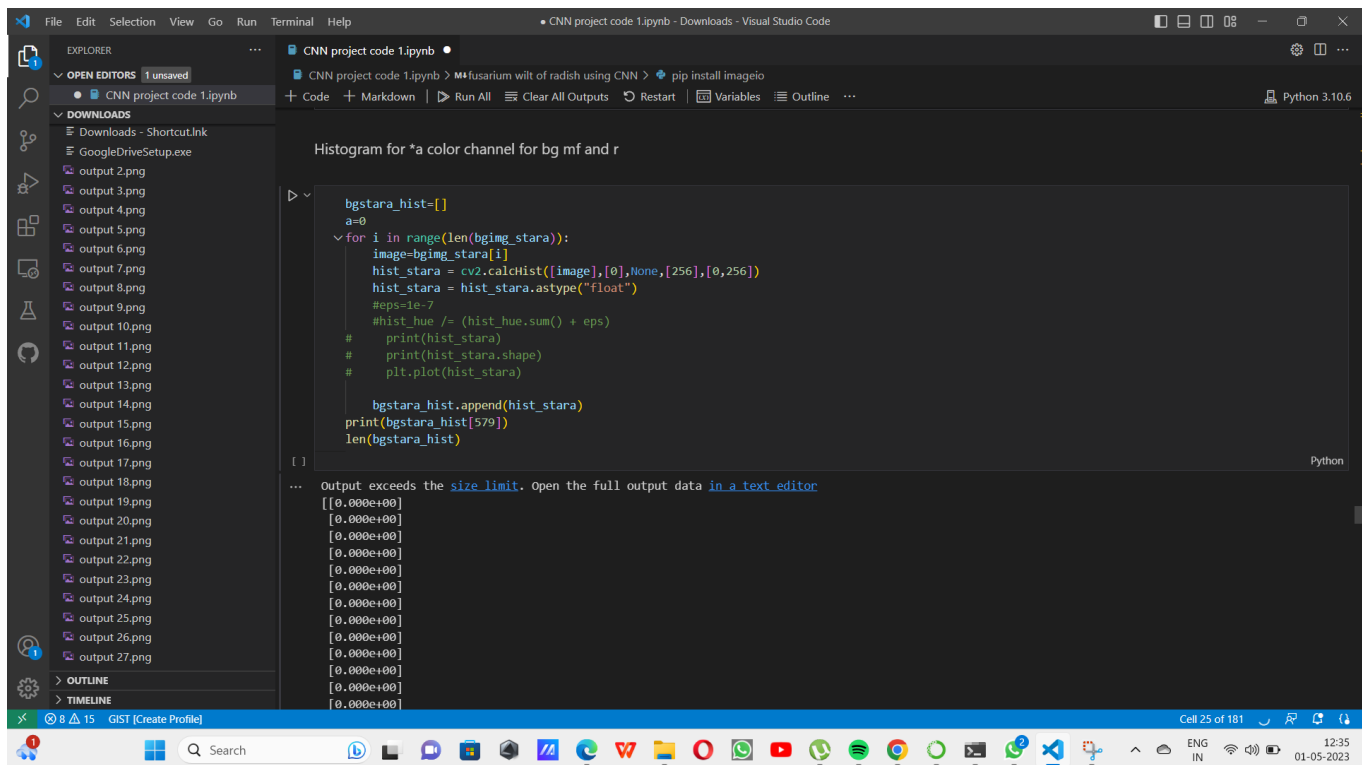


Figure 11

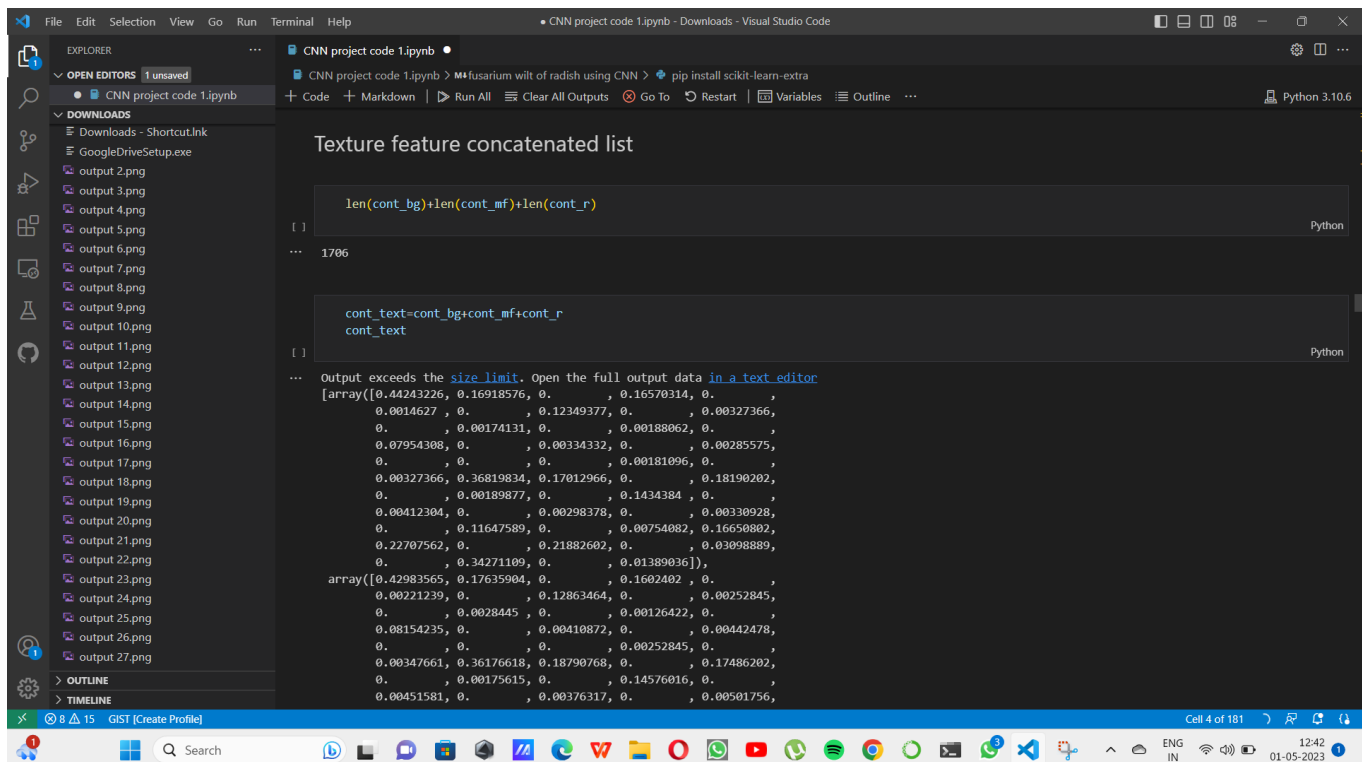


Figure 12

REFERENCE

1. Lee,H., Lee, W., D., Lee, G., & Lee, S. (2019). Development of a deep learning-based detection model for fusarium wilt of radish using convolutional neural network. *Computers and Electronics in Agriculture*, 156, 47-54.
2. He, X., Zhao, Z., Zhao, Y., & Wang, L.(2020).Fusarium wilt of radish detection using deep learning technology. *Information Processing in Agriculture*,7(2), 173-178.
3. Kim,M.J., Kim, Y., J., & Lee,J.H. (2019). Detection of fusarium wilt of radish using deep convolutional neural network. *Plant pathology journal*, 35(4),396-403.
4. Huang,X., Liu, X., & Wei, Z.(2019). Research on the detection of Radish fusarium wilt based on Deep learning. *Journal of Physics: Conference Series*, 1285, 012012.
5. Jiang, X., Liao, B., Peng, Y., Liu, J., Zhang, X., & Zhou, W. (2021). A method for identifying radish diseases based on convolutional neural networks. *Neural Computing and Applications*, 33(16), 11599-11612.
6. Chen, J., & Yang, X. (2020). Detection of fusarium wilt of radish based on deep convolutional neural network. *Journal of plant Protection*, 47(6), 1236-1242.
7. Jin,X., & joo, G J. (2018). A deep learning approach for the detection of fusarium wilt in radish leaves. *Sensors*, 18(10), 3432.
8. Khamphée, P., Baimai, S., Tongsiri, S., & Duangjai, S. (2020). Development of a CNN-based model for fusarium wilt detection in radish leaves using transfer learning. In 2020 5th international Conference on computer and communication Systems (ICCCS) (PP.274-278). IEEE.
9. Zhang, W., Wang, X., Yu, D., & Huang, X. (2021). Recognition of radish diseases based on transfer learning and deep learning. *Journal of plant Protection*, 48(1). 245-251.

APPENDIX-1

(SOURCE CODE)

```
import pandas as pd
from sklearn.cluster import KMeans
from sklearn_extra.cluster import KMedoids
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import silhouette_score
import os
import cv2
import matplotlib.pyplot as plt
bgimg_data=[]
for bgFile in bgFiles:
    bgimg_arr = cv2.imread(os.path.join(bgpath, bgFile))
    bgimg_data.append(bgimg_arr)
    #img_data.append(img_arr)
def show_img(path):

    """
    Reading the image
    """

    img = cv2.imread(path)
    return img

def gray_img(path):

    """
    Reading the image in the grayscale format
```

```

"""

img = cv2.imread(path, 0)
return img

def color2gray(img):
    img=cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
    return img

def plot_img(path, title = None):

    """
    Plotting the image using plt.imshow(),
    a title won't be added by default,
    but if it's needed you can add it as an argument
    """

    plt.imshow(gray_img(path), cmap = "gray")
    if title != None:
        plt.title(title)

plot_img('D:\\capstone project\\capstone research papers\\Drone agriculture imagery system for
Radish Wilt Disease\\datasetA\\bare ground\\10.png')

class LocalBinaryPatterns:

    def __init__(self, numPoints, radius):
        """
        store the number of points and radius
        """

```

```

self.numPoints = numPoints
self.radius = radius

def describe(self, image, eps=1e-7):

    """
    compute the Local Binary Pattern representation
    of the image, and then use the LBP representation
    to build the histogram of patterns
    """

    lbp = feature.local_binary_pattern(image, self.numPoints, self.radius, method="ror")
    (hist, _) = np.histogram(lbp.ravel(), bins=np.arange(0, self.numPoints + 3), range=(0,
self.numPoints + 2))

    # normalize the histogram
    hist = hist.astype("float")
    hist /= (hist.sum() + eps)

    # return the histogram of Local Binary Patterns
    return hist

# class LocalBinaryPatternsflat:

#     def __init__(self, numPoints, radius):
#         """
#         store the number of points and radius
#         """
#         self.numPoints = numPoints
#         self.radius = radius

```

```

# def describe(self, image, eps=1e-7):

# """
# compute the Local Binary Pattern representation
# of the image, and then use the LBP representation
# to build the histogram of patterns
# """

# lbp = feature.local_binary_pattern(image, self.numPoints,
# self.radius, method="ror")
# (hist, _) = np.histogram(lbp.ravel(),
# bins=np.arange(0, self.numPoints + 3),
# range=(0, self.numPoints + 2))

# # normalize the histogram
# hist = hist.astype("float")
# hist /= (hist.sum() + eps)

# # return the histogram of Local Binary Patterns
# return hist

lbpGen = LocalBinaryPatterns(24, 3)
bgimg_gray243=[]
bg243_cont=[]
for i in range(len(bgimg_data)):
    bgimg_gray243.append(color2gray(bgimg_data[i]))
    lbpHistogram243bg = lbpGen.describe(bgimg_gray243[i])
    bg243_cont.append(lbpHistogram243bg)
#instead of append use extend to get all the features together as a single numpy array

```

```

#print(lbpHistogram243bg.shape)
#print(lbpHistogram243bg)

#bgimg_data[10].shape
def contbg(num):
    cont=[]
    a=bg243_cont[num]
    b=bg162_cont[num]
    c=bg81_cont[num]
    cont=np.concatenate((cont,a), axis=None)
    cont=np.concatenate((cont,b), axis=None)
    cont=np.concatenate((cont,c), axis=None)
    return cont

# fusarium wilt of radish

import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator

# Define the directories for the training and validation datasets
train_dir = '/path/to/training/data'
val_dir = '/path/to/validation/data'

# Define the image data generators for the training and validation datasets
train_datagen = ImageDataGenerator(rescale=1./255)
val_datagen = ImageDataGenerator(rescale=1./255)

# Define the batch size and image size
batch_size = 32

```

```

img_size = (224, 224)

# Define the number of classes (healthy and unhealthy)
num_classes = 2

# Define the CNN model
model = tf.keras.models.Sequential([
    tf.keras.layers.Conv2D(32, (3,3), activation='relu', input_shape=(img_size[0], img_size[1], 3)),
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Conv2D(64, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Conv2D(128, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Conv2D(128, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dropout(0.5),
    tf.keras.layers.Dense(512, activation='relu'),
    tf.keras.layers.Dense(num_classes, activation='sigmoid')
])

# Compile the model
model.compile(loss='binary_crossentropy', optimizer='rmsprop', metrics=['accuracy'])

# Define the image data generators for the training and validation datasets
train_generator = train_datagen.flow_from_directory(train_dir, target_size=img_size,
batch_size=batch_size, class_mode='binary')

val_generator = val_datagen.flow_from_directory(val_dir, target_size=img_size,
batch_size=batch_size, class_mode='binary')

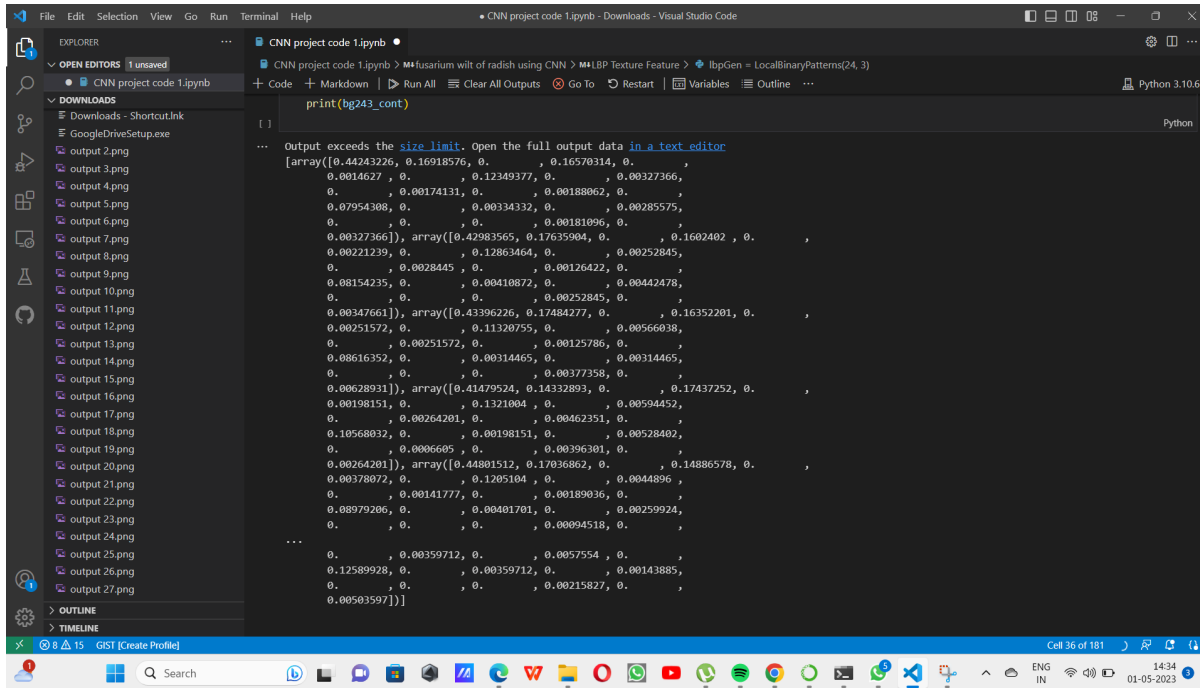
```

Fit the model to the training data

history = model.fit(train_generator, epochs=50, validation_data=val_generator)

APPENDIX-2

SCREENSHOT

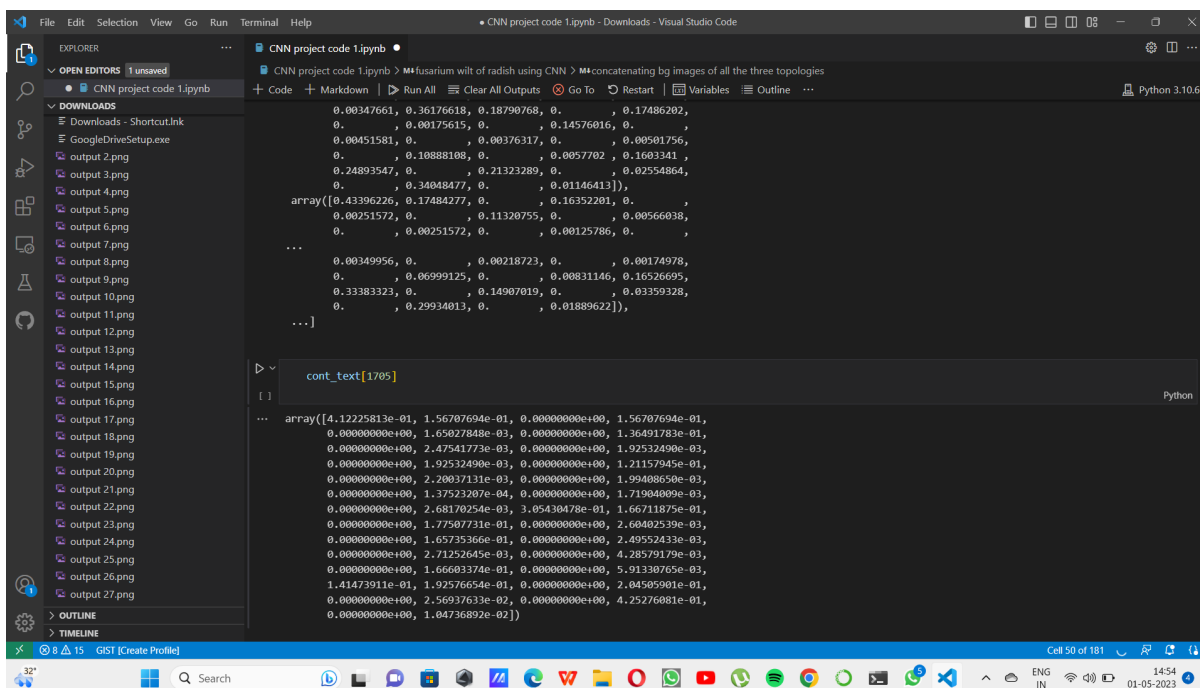


The screenshot shows a Visual Studio Code window with a Jupyter Notebook. The left sidebar displays the Explorer and Downloads panels. The main editor shows a code cell with the following content:

```
print(bg243_cont)

...
Output exceeds the size limit. Open the full output data in a text editor
[array([[0.44243226, 0.16918576, 0., 0.16570314, 0.,
0.0014627, 0., 0.12349377, 0., 0.00327366,
0., 0.00174131, 0., 0.00188062, 0.,
0.07954388, 0., 0.00334332, 0., 0.00285575,
0., 0., 0.00181096, 0.,
0.00377366]], array([[0.42983565, 0.17635904, 0., 0.1602402, 0.,
0.00221239, 0., 0.12863464, 0., 0.00252845,
0., 0.0028445, 0., 0.00126422, 0.,
0.08154235, 0., 0.00410872, 0., 0.00442478,
0., 0., 0.00252845, 0.,
0.00347661]], array([[0.43396226, 0.17484277, 0., 0.16352201, 0.,
0.00251572, 0., 0.11320755, 0., 0.00566038,
0., 0.00251572, 0., 0.00125786, 0.,
0.08616352, 0., 0.00314465, 0., 0.00314465,
0., 0., 0., 0.00377358, 0.,
0.00628931]], array([[0.41479524, 0.14332893, 0., 0.17437252, 0.,
0.00198151, 0., 0.1321004, 0., 0.00594452,
0., 0.00264201, 0., 0.00462351, 0.,
0.10568032, 0., 0.00198151, 0., 0.00528402,
0., 0.0006605, 0., 0.00396301, 0.,
0.00264201]], array([[0.44801512, 0.17036862, 0., 0.14886578, 0.,
0.00378072, 0., 0.1205104, 0., 0.0044896,
0., 0.00141777, 0., 0.00189036, 0.,
0.08979206, 0., 0.00401701, 0., 0.00259924,
0., 0., 0., 0.00094518, 0.,
0., 0., 0.00359712, 0., 0.0057554, 0.,
0.12589928, 0., 0.00359712, 0., 0.00143885,
0., 0., 0., 0.00215827, 0.,
0.00503597]]])

...
```



The screenshot shows a Visual Studio Code window with a Jupyter Notebook. The left sidebar displays the Explorer and Downloads panels. The main editor shows a code cell with the following content:

```
...
concatenating by images of all the three topologies

0.00347661, 0.36176618, 0.18790768, 0., 0.17486202,
0., 0.00175615, 0., 0.14576016, 0.,
0.00451581, 0., 0.00376317, 0., 0.00501756,
0., 0.10888108, 0., 0.0057702, 0.1603341,
0.24893547, 0., 0.21323289, 0., 0.02554864,
0., 0.34048477, 0., 0.01146413]],
array([[0.43396226, 0.17484277, 0., 0.16352201, 0.,
0.00251572, 0., 0.11320755, 0., 0.00566038,
0., 0.00251572, 0., 0.00125786, 0.,
0.00349956, 0., 0.00218723, 0., 0.00174978,
0., 0.06999125, 0., 0.00831146, 0.16526695,
0.33383323, 0., 0.14907019, 0., 0.03359328,
0., 0.29934013, 0., 0.01889622]],
...])

cont_text[1705]

...
array([[4.12225813e-01, 1.56707694e-01, 0.00000000e+00, 1.56707694e-01,
0.00000000e+00, 1.65027848e-03, 0.00000000e+00, 1.36491783e-01,
0.00000000e+00, 2.47541773e-03, 0.00000000e+00, 1.92532490e-03,
0.00000000e+00, 1.92532490e-03, 0.00000000e+00, 1.21157945e-01,
0.00000000e+00, 2.20037131e-03, 0.00000000e+00, 1.99408650e-03,
0.00000000e+00, 1.37523207e-04, 0.00000000e+00, 1.71904009e-03,
0.00000000e+00, 2.68170254e-03, 3.05430478e-01, 1.66711875e-01,
0.00000000e+00, 1.77507731e-01, 0.00000000e+00, 2.60402539e-03,
0.00000000e+00, 1.65735366e-01, 0.00000000e+00, 2.49552433e-03,
0.00000000e+00, 2.71252645e-03, 0.00000000e+00, 4.28579179e-03,
0.00000000e+00, 1.66603374e-01, 0.00000000e+00, 5.91330765e-03,
1.41473911e-01, 1.92576654e-01, 0.00000000e+00, 2.04505901e-01,
0.00000000e+00, 2.56937633e-02, 0.00000000e+00, 4.25276081e-01,
0.00000000e+00, 1.04736892e-02]])
```


The screenshot shows the Visual Studio Code interface with a Jupyter Notebook open. The notebook is titled "CNN project code 1.ipynb". The left sidebar shows the Explorer view with the file "CNN project code 1.ipynb" selected. The main editor area shows the notebook content, which includes a code cell with a large array output. The array is a 10x10 grid of floating-point numbers, representing the output of a CNN model. The output is truncated with "...". The status bar at the bottom shows the file is saved, the language is Python 3.10.6, and the current cell is 50 of 181.

The screenshot shows the Visual Studio Code interface with a Jupyter Notebook open. The notebook is titled "CNN project code 1.ipynb". The left sidebar shows the Explorer view with the file "CNN project code 1.ipynb" selected. The main editor area shows the notebook content, which includes a code cell with a large array output. The array is a 10x10 grid of floating-point numbers, representing the output of a CNN model. The output is truncated with "...". The status bar at the bottom shows the file is saved, the language is Python 3.10.6, and the current cell is 50 of 181.

