

Potato Disease Classification

Project report submitted in partial fulfillment of the requirement
for the degree of Bachelor of Technology in

Computer Science and Engineering

By- Harsh Malik(201379)

Dhruv Parasher(201503)

Under the Supervision of Dr. Hari Singh



Department of Computer Science & Engineering and
Information Technology

Jaypee University Of Information Technology , Solan
Himachal Pradesh

CERTIFICATE

CANDIDATE’S DECLARATION

I hereby declare that the work presented in this report entitled “**Potato Disease Classification**” in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology in Computer Science and Engineering/Information Technology** submitted in the department of Computer Science & Engineering and Information Technology, Jaypee University of Information Technology Waknaghat is an authentic record of my own work carried out over a period from July 2023 to December 2023 under the supervision of **Dr. Hari Singh** Assistant Professor(SG) Computer Science and Engineering .

The matter embodied in the report has not been submitted for the award of any other degree or diploma.

Harsh Malik (201379)

Dhruv Parasher (201503)

This is to certify that the above statement made by the candidate is true to the best of my knowledge.

(Supervisor Signature)

Dr.Hari Singh

Assistant Professor(SG)

Computer Science and Engineering

Dated:28/11/2023

PLAGIARISM CERTIFICATE

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY, WAKNAGHAT PLAGIARISM VERIFICATION REPORT

Date: 01/12/2023
 Type of Document (Tick): ☐ PhD Thesis ☒ M.Tech/M.Sc. Dissertation ☐ B.Tech./B.Sc./BBA/Other
 Name: Harsh Malik, Dhruv Parasher Department: CSE Enrolment No. 201379
 Contact No. 7678147359 E-mail: 201379@juitsoan.in
 Name of the Supervisor: Dr. Hari Singh
 Title of the Thesis/Dissertation/Project Report/Paper (In Capital letters):
Potato Disease Classification using CNN

UNDERTAKING

I undertake that I am aware of the plagiarism related norms/ regulations, if I found guilty of any plagiarism and copyright violations in the above thesis/report even after award of degree, the University reserves the rights to withdraw/revoke my degree/report. Kindly allow me to avail Plagiarism verification report for the document mentioned above.

- Total No. of Pages = 57
- Total No. of Preliminary pages = 7
- Total No. of pages accommodate bibliography/references = 2

(Signature of Student)

FOR DEPARTMENT USE

We have checked the thesis/report as per norms and found Similarity Index at 16 (%). Therefore, we are forwarding the complete thesis/report for final plagiarism check. The plagiarism verification report may be handed over to the candidate.

(Signature of Guide/Supervisor)

Signature of HOD

FOR LRC USE

The above document was scanned for plagiarism check. The outcome of the same is reported below:

Copy Received on	Excluded	Similarity Index (%)	Abstract & Chapters Details	
Report Generated on	<ul style="list-style-type: none"> All Preliminary Pages Bibliography/Images/Quotes 14 Words String 		Word Counts	
			Character Counts	
		Submission ID	Page counts	
			File Size	

Checked by
Name & Signature

Librarian

Please send your complete Thesis/Report in (PDF) & DOC (Word File) through your Supervisor/Guide at plagcheck.juit@gmail.com

ACKNOWLEDGEMENT

First and foremost, I want to give God the highest praise for His heavenly grace, which enabled us to successfully finish the project work. My mentor, **Dr. Hari Singh, Assistant Professor (SG), Department of CSE Jaypee University of Information Technology** has my deepest gratitude. My mentor has a wealth of knowledge and a genuine interest in the "Research Area" needed to complete this assignment. This project was made possible by his never-ending patience, academic leadership, constant encouragement, frequent and vigorous supervision, constructive criticism, insightful counsel, reviewing several subpar versions and revising them at all levels. It is my regarded joy to introduce this project and earnestly thank each and every individual who helped me in this project.

I express my earnest gratitude to **Jaypee University of Information Technology** for giving an open door and such a decent learning climate. I express my sincere gratitude to Jaypee University of Information Technology, Solan for offering help for everything and for giving productive analysis and support which prepared us to the fruitful culmination of the project.

I'm incredibly grateful to **Dr. Hari Singh(SG)**, (mentor for the project and Assistant professor (SG)) for his important direction and backing. I'm likewise thankful to the subjects of this review for their collaboration and interest. Last however not the least I thank god and my parents for every one of the endowments.

(Student Signature)

(Student Signature)

Student Name

Harsh Malik

Dhruv Parasher

ABSTRACT

Potatoes constitute a vital global crop, yet they are susceptible to various diseases that can significantly impact both yield and quality. Timely detection and identification of these diseases are crucial for implementing effective management strategies to mitigate crop losses. Artificial intelligence (AI) techniques present a promising approach for the accurate and efficient diagnosis of potato diseases.

One of the frequently employed AI algorithms in the classification of potato diseases is Convolutional Neural Networks (CNNs). CNNs represent a type of deep learning algorithm capable of extracting intricate features from images, rendering them well-suited for disease detection tasks. Numerous studies have documented elevated accuracy rates in the classification of potato diseases through the application of CNNs, with some achieving precision levels nearing 100%.

To assess the presentation of the AI models, different measurements like exactness, accuracy, review, and F1-score are utilized. Cross-approval is additionally used to test the power of the models against overfitting and to improve the hyperparameters of the calculations.

TABLE OF CONTENT

SR NO.	TITLE	SECTIONS	PAGE NO.
1.	Declaration		ii
2.	Certificate		iii
3.	Acknowledgment		iv
4.	Abstract		v
5.	CHAPTER 1 : INTRODUCTION	1.1 Introduction 1.2 Problem Statement 1.3 Objectives 1.4 Methodology 1.5 Organization of Project Report	10-17
6.	CHAPTER 2 : LITERATURE SURVEY	2.1 Overview of Relevant Literature 2.2 Key Gaps in the Literature	18-22
7.	CHAPTER 3 : SYSTEM DEVELOPMENT	3.1 Requirements and Analysis 3.2 Project Architecture and Dataset 3.3 Data Preparation 3.4 Algorithm/Pseudocode 3.5 Key Challenges	23-42
8.	CHAPTER 4 :	4.1 Experiment and Analysis	43-49

	TESTING		
9.	CHAPTER 5 : RESULTS AND EVALUATION	5.1 Results 5.2 Comparison with Existing Solutions	50-51
10.	CHAPTER 6 : CONCLUSIONS AND FUTURE SCOPE	6.1 Conclusion 6.2 Future Scope 6.3 Applications 6.4 Limitations	52-53
11.	References and Appendices		54-55

LIST OF ABBREVIATIONS

- **UI - USER INTERFACE**
- **HD - HIGH DIMENSIONAL**
- **FC- FULLY CONNECTED**
- **AI - ARTIFICIAL INTELLIGENCE**
- **CNN - CONVOLUTIONAL NEURAL NETWORK**
- **NFR - NON FUNCTIONAL REQUIREMENTS**
- **RGB - RED GREEN BLACK**
- **SVM - SUPPORT VECTOR MACHINE**
- **KNN - K NEAREST NEIGHBOUR**
- **ANN - ARTIFICIAL NEURAL NETWORK**
- **DPI - DOTS PER INCH**
- **UML - UNIFIED MODELING LANGUAGE**

LIST OF FIGURES

Figure Title	Page Number
Fig 1.1: Leaf species with sickness	11
Fig 1.2: Algorithm processing	13
Fig 1.3: Feature Extraction	15
Fig 1.4: Architecture	16
Fig 1.5: Flow Chart	24
Fig 3.1: Type of Leaves	25
Fig 3.2: Use Case Diagram	25
Fig 3.3: CNN Functioning	29
Fig 3.4: Pooling Diagram	31
Fig 3.5: Padding	32
Fig 3.6: Activation Functions	33
Fig 3.7: Pooling Numeric Example	33

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION

As we are aware, potatoes rank among the most widely consumed crops. To ensure continued robust crop production, it is imperative to identify factors that may impede it. *Phytophthora infestans* and *Alternaria solani* are two pathogens responsible for significant production losses in potatoes, causing diseases known as late blight and early blight, respectively. Early detection of these diseases would enable the implementation of preventive measures, thereby reducing financial and yield losses. Traditional methods for detecting and identifying plant diseases, such as expert naked-eye monitoring, have been common in recent decades. However, this approach becomes impractical in many instances due to a shortage of experts in remote farming locations and lengthy processing times.

Consequently, the utilization of image processing technologies has emerged as an excellent approach for continuous plant health monitoring and early disease diagnosis. Therefore, the aim of this project is to develop a classification methodology using a simple convolutional neural network to detect diseases by inputting potato leaf images.

In our project we used 54,306 images of 14 species. Create a visual representation showcasing each disease paired with its corresponding crop, utilizing the information readily available through the PlantVillage initiative, which encompasses 26 distinct illnesses.



Fig 1.1: Leaf species with sickness

1.2 PROBLEM STATEMENT

In India, where over 70% of the population depends on agriculture, identifying plant diseases is crucial for averting crop losses. Manual detection of plant diseases is labor-intensive and time-consuming, requiring extensive knowledge and expertise. To address this challenge, our approach for plant disease identification from leaf images, using image processing and artificial intelligence (AI) models. Image processing extracts relevant information from images, while AI, a subset of human intelligence, interprets and processes data to perform specific tasks. Our method analyzes various image features such as leaf color, damage extent, location, and texture to achieve optimal accuracy in identifying different plant diseases. This automated approach proves to be more cost-effective and efficient compared to manual methods, providing a computational solution that requires less investment and expertise while delivering accurate results in plant disease detection.

Project statement is to classify whether a plant is healthy or not by using an image of a potato leaf. It is a multi-class classification problem. Following are the three classes:

- Late Blight- Potato leaves are more deteriorated.
- Early Blight- Potato leaves are in the early stage of disease.
- Healthy- Leaves are healthy

1.3 OBJECTIVES

The Indian economy heavily relies on the agricultural sector, contributing 18% to the country's GDP, with approximately 60% of the population engaged in this field. To enhance crop production, new technologies are imperative. Early detection of diseases is crucial, allowing for the implementation of preventive measures and, consequently, an increase in crop production and revenue. Manual monitoring is time-consuming and demands specialized expertise. Utilizing image processing techniques for accurate plant classification can significantly save time and resources. Therefore, the aim of this project is to employ Convolutional Neural Networks (CNN) for image classification and develop a web application that takes an image as input, classifying it based on a pre-trained model.

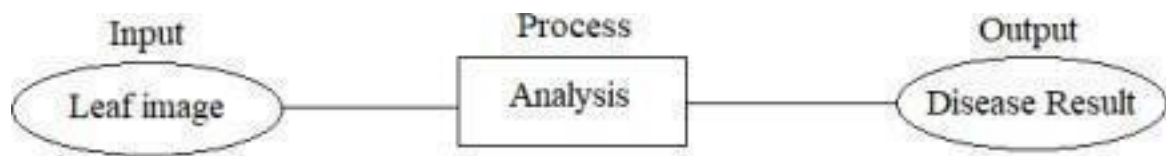


Fig 1.2: Algorithm processing

1.4 METHODOLOGY

CNN is a subset of Deep Neural Networks (DNN) specifically designed for identifying and categorizing distinctive features in visual imagery, as depicted in Fig1.3. The CNN comprises two fundamental components:

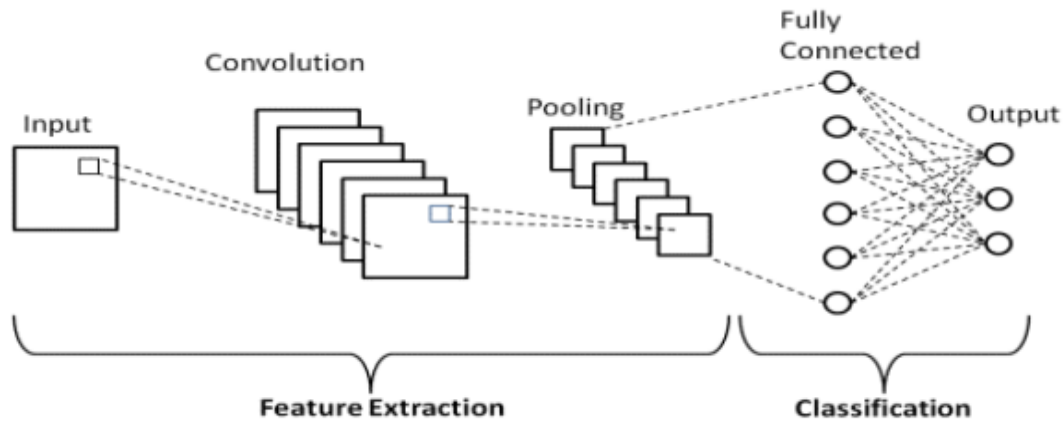


Fig1.3

Feature Extraction: This is a feature which helps in distinguishing amongst different features of the data.

FC Layer: Used to identify raw feature of an image.

The CNN architecture encompasses three major types of layers: convolutional layers, pooling layers, and fully-connected (FC) layers. Additionally, there are two supplementary layers, namely the dropout layer and the activation layer. When these layers are integrated, they form the complete CNN architecture as shown in figure 1.4.

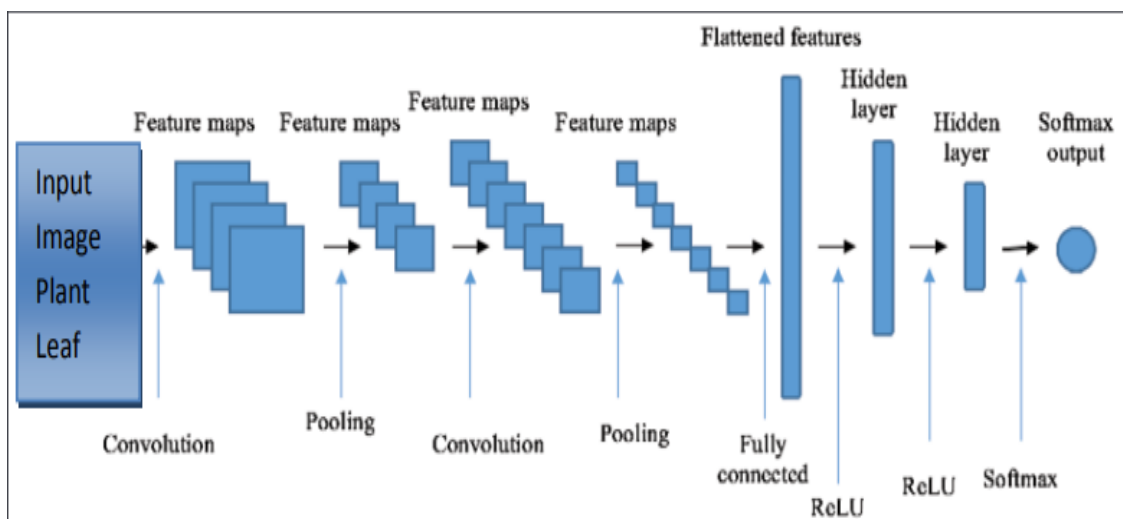


Fig 1.4

Convolutional Layer: The initial layer extracts diverse features from the image through a convolutional process involving a filter of a specified size $N \times N$ and the input image. Output is in form of a feature map, which serves as input for the subsequent layer.

Pooling Layer: This layer is responsible for diminishing the dimensions of the convolved feature map, mitigating computational costs. Two types of pooling exist: max pooling and average pooling.

Dropout: This layer selectively drops certain neurons to address the issue of overfitting.

Activation Functions: This layer holds significance in decision-making, determining which information is relevant for the forward direction and which is not.

The model building flowchart includes all the major steps our model will include as shown in figure 1.5.

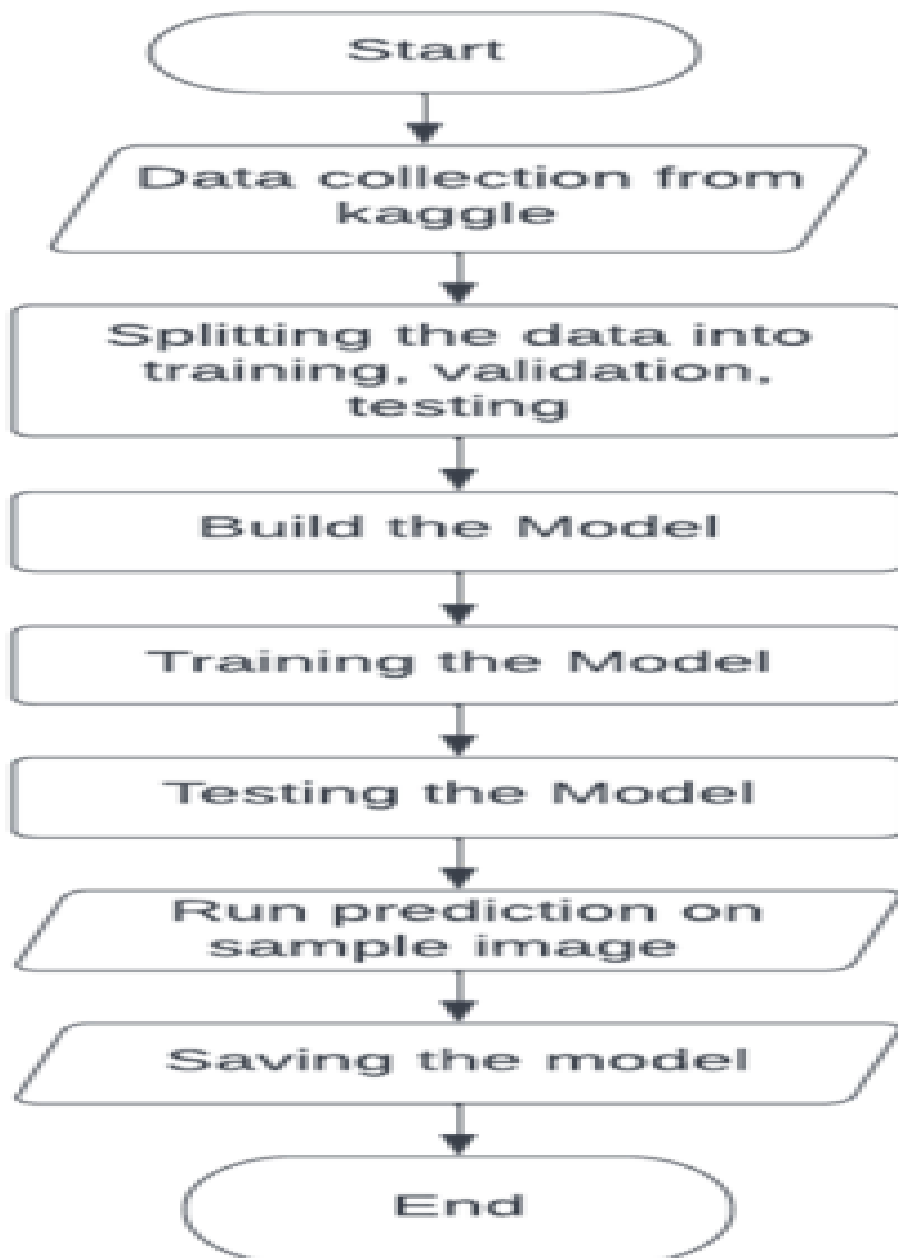


Fig 1.5: Model Building Flowchart

1.5 ORGANIZATION OF PROJECT REPORT

Introduction to Project Structure:

The project report aims to provide a thorough exploration of the implementation and outcomes of potato disease classification using Convolutional Neural Networks (CNN). The organizational structure ensures a logical flow, offering a comprehensive understanding of the project's objectives and findings.

Section Breakdown:

a. Introduction:

Introduces the background, objectives, and scope of the project in potato disease classification using CNN.

b. Literature Review:

Examines existing research in plant disease classification and CNN applications, establishing the theoretical foundation for the proposed approach.

c. Methodology:

Details the steps involved in implementing the CNN architecture, dataset preparation, and the training process for potato disease classification.

d. Model Architecture:

Provides an in-depth examination of the CNN architecture utilized, accompanied by visualizations of the network's components.

e. Discussion:

Interprets results, compares findings with existing literature, and addresses limitations and potential directions for future research in potato disease classification.

f. Conclusion:

Summarizes key findings, contributions, and concluding thoughts on the outcomes of potato disease classification using CNN.

g. Results:

Presents quantitative and qualitative results, including accuracy metrics and visual comparisons of disease classification outcomes.

CHAPTER - 2

2.1 OVERVIEW OF RELEVANT LITERATURE

TABLE 1 : LITERATURE REVIEW SURVEY

S. No.	Paper Title [Cite]	Journal/Conference(Year)	Tools/Techniques/Dataset	Results	Limitations
1.	Plant Disease Detection and Classification Method Based on the Optimized Lightweight YOLOv5 Model [1]	College of Mechanical and Electrical Engineering, Qingdao Agricultural University, Qingdao 266109, China (2022)	Faster R-CNN, VGG16, SSD, Efficient, YOLOv4, YOLOv5, optimized lightweight YOLOv5, Used data from Plant Village data.	The accuracy reached 90.26% and 92.57%	The positioning error of the detection frame, and the positioning error in the detection of small targets.
2.	Plant Disease Detection and Classification by Deep Learning [2]	IEEE Access, vol. 9, pp. 56683-56698, 2021, doi:10.1109/ACCESS.2021.3069646. (2021)	SVM, K-means, New CNN model ARNet based on the combination of attention and residual ideas.	The accuracy of the two methods reached 94.71% and 98.32%, respectively .	Lack of diverse and well-annotated datasets
3.	Comparison of Plant Leaf Classification Using Modified Alex Net and Support Vector Machine [3]	Traitement du Signal Vol. 38, No. 1, February, 2021, pp. 79-87 (2021)	SVM with linear kernel and radial function kernel, Alex Net, and the data of 3200 images of 9 different plants used.	For 70% of training data accuracy is 89.69% and for 90% data is 89.38%.	Comparison with other architectures isn't mentioned
4.	Plant Disease Detection Using CNN	IEEE 2020 IEEE Applied Signal Processing Conference(ASP	CNN with 4 Layers is used (Convolutional , Pooling, Max	The accuracy percentage on the test	There is no comparative analysis with other existing

	[4]	CON) (2020)	Pooling & Fully Connected layer), total of 3000 images has been provided	set is 88.80% with no overfitting	architectures.
5.	Potato Leaf Disease Classification Using Deep Learning Approach [5]	2020 International Electronic Symposium (IES) (2020)	Modern CNN architectural models such as AlexNet, VGG, GoogLeNet, ResNet. Dataset is taken from Kaggle.	VGG Network has potential for studying effective features for image classification of leaf diseases with accuracy of 91%.	Noise in the image will result in poor classification results.
6.	CNN based Disease Detection Approach on Potato Leaves [6]	3rd International Conference on Intelligent Sustainable Systems (ICISS) (2020)	Techniques are AlexNet, VggNet, ResNet, LeNet & Sequential models. For the dataset -3000 images were collected and merged.	The offered model distinguishes the diseased & healthy plants appropriately with a precision of 97%.	The resolution of certain images is too low, some are hardly detected as affected and unaffected.
7.	Comparison of Artificial Intelligence Algorithms in Plant Disease Prediction [7]	Revue d'Intelligence Artificielle (2019)	CNN, RNN, ANN, SVM, KNN, Squared error loss function is used for training.	ANN outperforms all the other algorithms compared in this paper with accuracy of 90.79%	The accuracy of potato disease classification models can be affected by environmental conditions, such as lighting, temperature, and humidity
8.	Disease	International	Model is	The	The model

	Detection and Classification in Agricultural Plants Using Convolutional Neural Networks [8]	conference on SPIN (2019)	created using 4 convolution layers followed by a ReLU function and pooling layer. Apple and tomato leaf image dataset containing 3663 images.	achieved accuracy is 88.7 with minimum number of parameters i.e.. 45K when compared to other existing models.	overfits as there is a gap between the training and validation curves.
9.	Detection of Potato Diseases Using Image Segmentation and Multiclass Support Vector Machine[9]	Department of Electrical and Computer Engineering University of Saskatchewan Saskatoon, Canada (2017)	SVM is used for image classification, Gray Level Co-occurrence Matrix was used for extracting statistical texture, and a database of 300 images of potato leaves.	100 healthy leaves and 200 diseased leaves, 93.7% accuracy was achieved.	High computational cost.
10.	Application of neural networks to image recognition of plant diseases[10]	Department of Plant Pathology, China Agricultural University, Beijing 100193, China (2012)	185 digital images of plant diseases to which image compression, image cutting and image enhancement were applied. BP networks, RBF neural networks, GRNNs and PNNs are used as the classifiers .	Fitting accuracy $\geq 75\%$ and prediction accuracy $\geq 75\%$	Lack of real-world validation

2.2 KEY GAPS IN THE LITERATURE

Although the review of the literature offers insightful information on different approaches to the categorization of potato diseases, certain significant gaps and potential research areas are highlighted. Among the gaps found are:

- **Localization Accuracy Issues:**

This restriction implies that precisely localizing and identifying the regions of interest that is, the areas impacted by the disease within the plant photos can provide difficulties. It suggests that the detection system might have trouble with accuracy, particularly when dealing with minor or complex patterns linked to plant illnesses.

- **Lack of diverse and well-annotated datasets:**

In the world of machine learning , one major difficulty is the lack of diverse and well-annotated datasets. A small dataset could make it more difficult for the model to generalize to other situations and illnesses. Additionally, accurate model training depends heavily on the quality of annotations. Incomplete or erroneous annotations, or a lack of diversity in the dataset, might negatively impact the model's performance and dependability in practical settings.

- **High computational cost:**

This restriction highlights the artificial intelligence algorithms' high resource requirements for detecting plant diseases. High computational costs can be a major disadvantage, especially when deploying in real-time applications or environments with limited resources. It might restrict the suggested solution's scalability and usefulness, particularly in situations with constrained computer resources.

- **Overfitting Gap:**

When a model learns the training set too well, it is said to be overfitting and loses its ability to generalize to new, unobserved data. The model may perform well on training data but have difficulty generalizing to validation data if there is a gap between the training and validation curves. This constraint suggests that the model may be very intricate, identifying noise in the training data instead of the fundamental patterns, and modifications might be required to enhance generalization.

- **Low-Resolution Detection Challenge:**

Low-quality photos could not provide the information required for an appropriate diagnosis of a disease. Image resolution is essential for catching minute details. False positives and false negatives may arise from images that are rarely identified as impacted or unaffected. One way to get over this limitation might be to investigate methods for enhancing characteristics in low-quality photos or to improve the resolution of existing photographs.

- **Image Noise impact:**

Random fluctuations in pixel values, or "image noise," can impede the model's capacity to correctly categorize illnesses. Noise may originate from a number of causes, including visual artifacts or defective sensors. It can be essential to employ noise-resistant algorithms or strong preprocessing methods in order to decrease the impact of noise and enhance classification performance overall.

CHAPTER - 3

SYSTEM DEVELOPMENT

This chapter outlines the project construction procedure, encompassing model and system development discussions. It covers the dataset details for model training, explores the Python libraries employed, and provides insights into the explained model architecture. The theory behind the convolutional neural network is elucidated by detailing each neural network component and various layers used.

3.1 REQUIREMENTS AND ANALYSIS

Language used - Python

NumPy: Numpy usually carries out all the mathematical computation and deals with different dimensions of array.

Pandas: Pandas is used to play with the datasets and to know various insights about the dataset.

scikit-learn: sci-kit learn is the most used library in the field of machine learning it have packages of all the algorithms predefined.

Matplotlib: It is the most used Python Visualization Tool in the industry its also has some predefined plots which makes the visualization smoother

Keras, TensorFlow: Keras is the only python library which deals with neural networks smoothly.

3.2 PROJECT ARCHITECTURE AND DATASET

For this project, images were sourced from Kaggle , an open repository, specifically utilizing the PlantVillage dataset. This dataset encompasses over 55,000 images of plant leaves representing various species, including both healthy and infected leaves. The variety of leaf images across different classes is not even, ranging from 152 to 1000 images per class. Our focus was on potato leaves, and we exclusively utilized images from the PlantVillage dataset. All images were standardized by resizing them to 256x256 pixels, encompassing healthy leaves, early blight, and late blight leaves. Sample images from each class can be observed in Figure 3.1.



Fig 3.1.

3.2.1. USE CASE DIAGRAM

The module begins by extracting features from the inputted image of the leaf. Subsequently, the CNN model is employed, comparing these features to a pre-trained dataset. Following this, the module undergoes a dense CNN process to extract leaf features independently. Finally, the module assesses whether the plant leaf is infected with a disease.

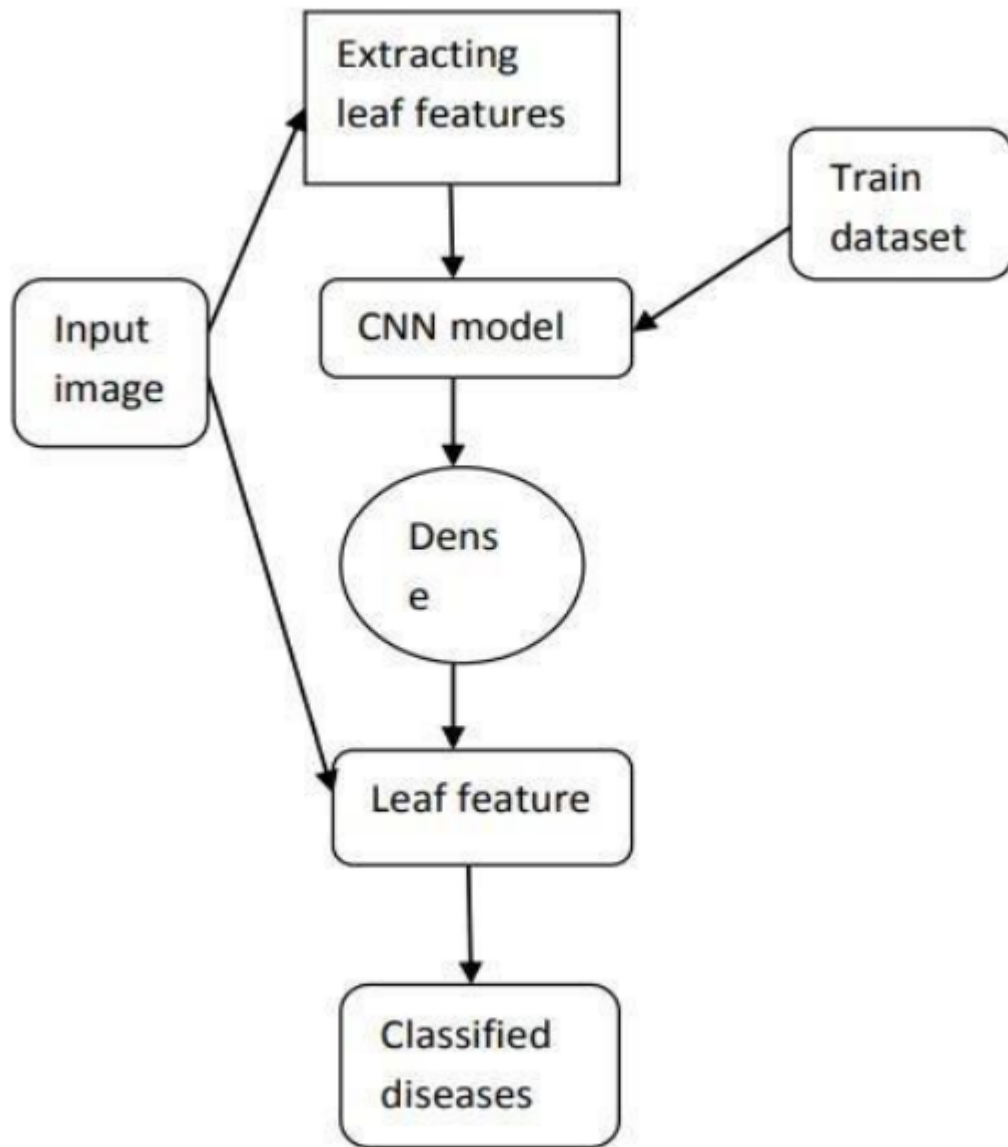


Fig 3.2: User Case Diagram

3.3 DATA PREPARATION

3.3.1 Image Acquisition

Any supervised machine learning project starts with data collection process. There are basically three steps to collect dataset:

- 1) Collect and annotate data on your own.
- 2) Write web scraping scripts to collect images from the internet.
- 3) Buy data from third party vendors or use public repositories such as Kaggle.
- 4) Dataset is taken from Kaggle repository. Dataset consists of images belonging to three different classes. Below is the link for dataset:
<https://www.kaggle.com/arjuntejaswi/plant-village>

3.3.2 Image Processing

When we gather images directly from the field, they often come with natural elements like dust, spores, and water spots, introducing unwanted noise. The purpose of data pre-processing is to clean up this noise in the image by adjusting pixel values, ultimately enhancing the overall image quality.

The primary goal of pre-processing is twofold: first, to eliminate undesirable image data, and second, to enhance critical image features. This involves converting the image from RGB to grayscale, resizing it, and applying median filtering. In this process, the color variation in the image is completely transformed into grayscale to ensure device independence. Subsequently, the image is resized to a standard size of 256x256, contributing to an improved and standardized representation.

3.3.3 Feature Extraction

In the process of potato disease classification using Convolutional Neural Networks (CNN), feature extraction involves systematically identifying and capturing distinctive patterns from input images of potato leaves. CNN's convolutional layers analyze the images to recognize edges, textures, and shapes, while pooling layers reduce dimensionality. The flattened features are then fed into dense layers, enabling the network to understand intricate relationships and patterns. The output layer provides probabilities for different disease categories. Through training on labeled datasets, the CNN learns to optimize its parameters, enhancing its ability to accurately classify potato leaves as healthy or affected by diseases like early blight or late blight. This hierarchical feature extraction approach empowers the model to automatically discern relevant patterns, contributing to effective disease classification.

3.4 Algorithms/Pseudocode of the Project.

Constants:

```
BATCH_SIZE = ...
```

```
IMAGE_SIZE = ...
```

```
CHANNELS = ...
```

```
n_classes = 3
```

Model Definition:

```
model = Sequential()
```

Preprocessing Layer:

```
model.add(resize_and_rescale)
```

Convolutional Layers with Max Pooling:

```
model.add(Conv2D(32, kernel_size=(3, 3), activation='relu',  
input_shape=(BATCH_SIZE, IMAGE_SIZE, IMAGE_SIZE,  
CHANNELS)))
```

```
model.add(MaxPooling2D((2, 2)))
```

```
model.add(Conv2D(64, kernel_size=(3, 3), activation='relu'))
```

```
model.add(MaxPooling2D((2, 2)))
```

```
model.add(Conv2D(64, kernel_size=(3, 3), activation='relu'))
```

```
model.add(MaxPooling2D((2, 2)))
```

```
model.add(Conv2D(64, (3, 3), activation='relu'))
```

```
model.add(MaxPooling2D((2, 2)))
```

```
model.add(Conv2D(64, (3, 3), activation='relu'))
```

```
model.add(MaxPooling2D((2, 2)))
```

```
model.add(Conv2D(64, (3, 3), activation='relu'))
```

This pseudocode outlines the construction of a convolutional neural network (CNN) for image classification. It includes constants for configuration, layer definitions, and the building of the model with specified input shapes.

3.4.1 What are Convolutional Neural Networks?

Convolutional Neural Networks (CNNs) represent a specialized category of neural networks renowned for their remarkable effectiveness in image categorization. Their

incorporation of additional layers, such as convolution and pooling, makes them particularly well-suited for applications in computer vision, notably excelling in tasks like face recognition and object detection. Typically comprised of multiple convolutional layers, these networks conclude with dense layers that resemble those found in conventional neural networks. A distinctive feature of CNNs lies in their adept exploitation of the two-dimensional structure inherent in images, utilizing tied weights and local connections. Subsequent to convolutional layers, max or mean pooling layers are employed to extract salient features. The key advantage of CNNs over regular neural networks lies in their reduced parameter count, translating to a more straightforward training process due to the fewer parameters requiring adjustment. This inherent simplicity makes CNNs easier to train in comparison to their neural network counterparts.

Overview of different operations in CNN in figure 3.3:



fig 3.3

As depicted in the figure above, the CNN architecture comprises sequential operations. This straightforward CNN includes a convolutional layer where a kernel or filter is utilized to identify vertical or horizontal edges, resumed by the application of the

Rectified Linear Unit (ReLU) activation function. The subsequent layer is a max-pooling layer, employing a kernel to extract the most prominent features from the image. This is succeeded by fully connected dense layers. The model's accuracy is determined by the loss function during forward propagation. Subsequently, in back propagation, the weights, filters, and bias are updated based on the loss value, employing methods such as gradient descent, AdaDelta, or Adam.

3.4.2 Convolution

In this process, a diminutive tensor, termed a kernel or filter, is applied to traverse an input image. The primary objective of this operation is to extract crucial features from the image. Subsequent to this operation, a feature map is generated, computed through element-wise multiplication of each value in the image and the kernel. This resulting feature map encapsulates the vertical and horizontal edges, which are subsequently amplified in the subsequent step.

Figure showing convolution operation

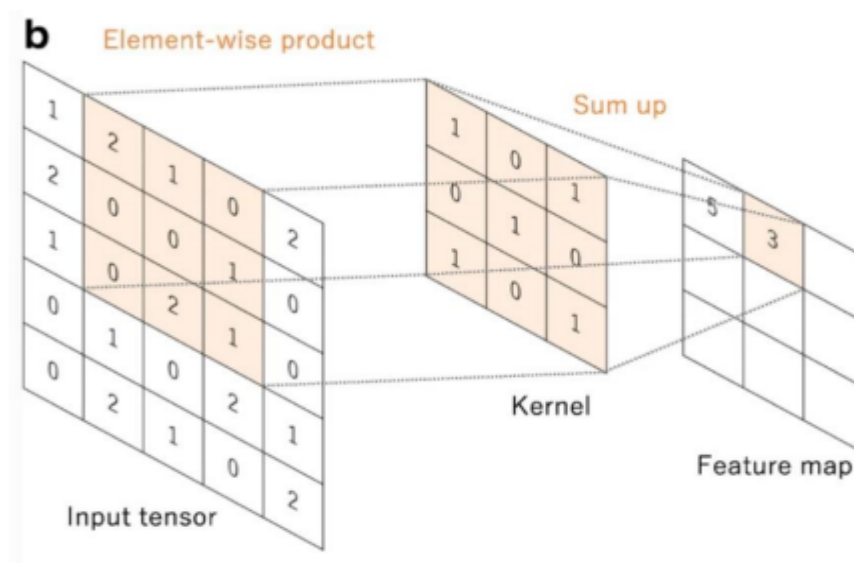
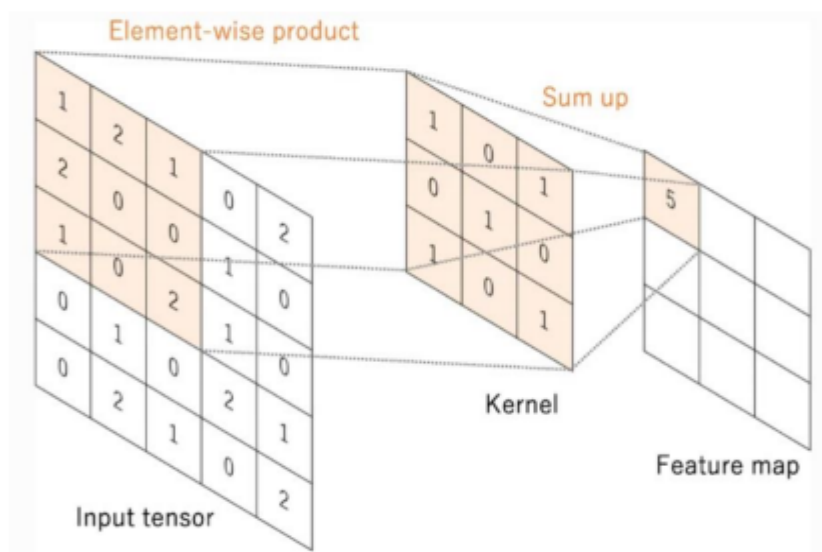


Fig 3.4

Above figure 3.4 is showing a convolution operation on a 5x5 image. Filter size is 3x3 without padding and stride equal to 1. If there is no padding then it will reduce its size as shown in the figure. To maintain the same image size padding is used which will add extra pixels across the image to maintain its size. Formula to calculate image size: $(N + 2P - F) / S + 1$ Where N = image size P = padding F = kernel size S = stride

Figure with padding:

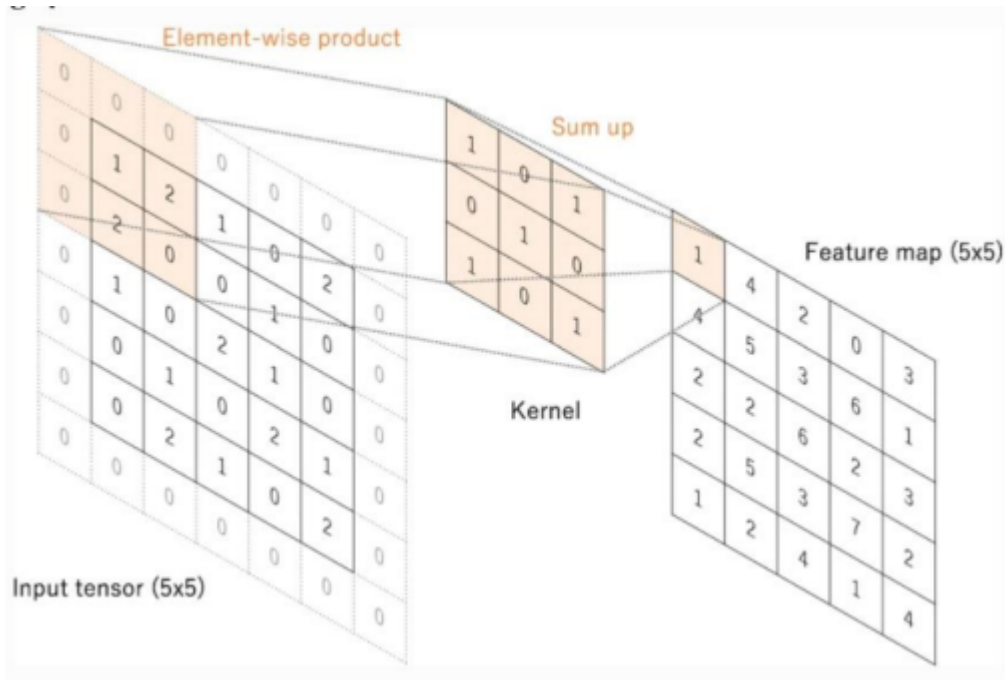


Fig 3.5

3.4.3 Activation Function

Feature maps which are obtained after applying convolution are passed through an activation function. These activation functions are non-linear to introduce non-linearity in the network. Activation functions such as sigmoid, tangent are not used in hidden layers as it may result in vanishing gradient problems. Highly used activation function is the Relu function. $F(x) = \max(0, x)$

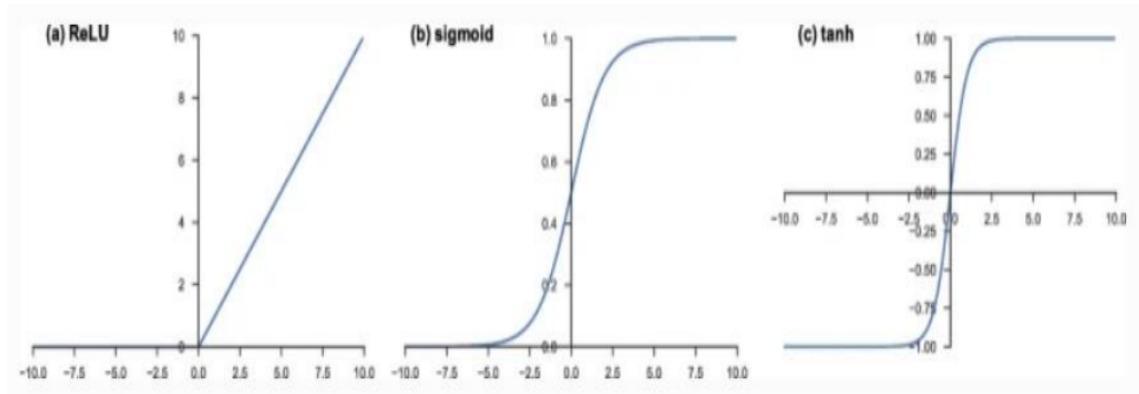


Fig 3.6

3.4.4 Pooling Layer

The Pooling layer is responsible for reducing the spatial dimensions of the convolved component. It proves beneficial in discarding dominant aspects that are rotationally and positionally invariant, thereby preserving the overall orientation during model training. Two types of pooling methods are commonly used:

Max Pooling: This method returns the maximum value from the section of the image covered by the pool.

Average Pooling: Conversely, this method returns the average of all the values from the section of the image covered by the pool.

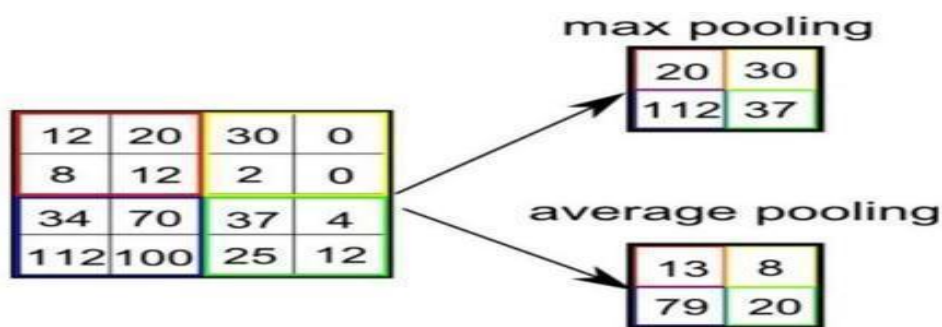


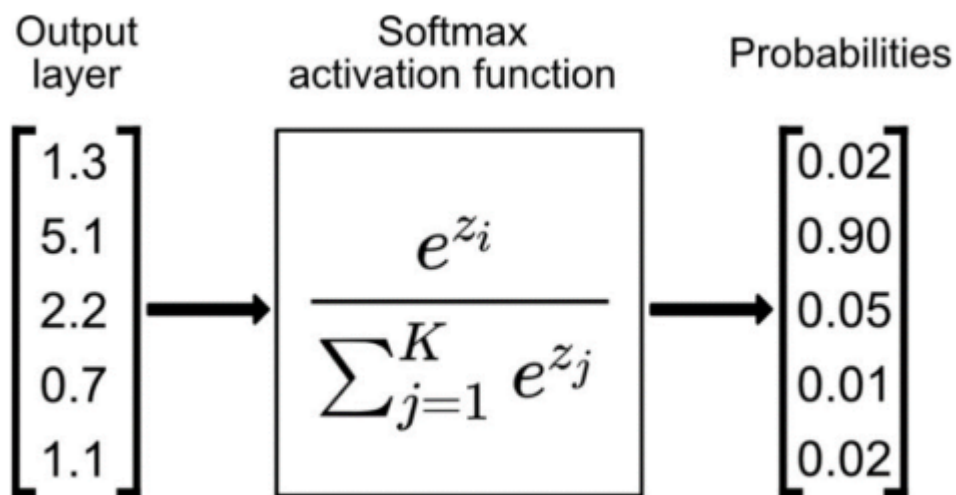
Fig3.7

3.4.5 Fully Connected Layer

Following the acquisition of feature maps from the max-pooling layer, these maps undergo a process of flattening and are subsequently transmitted to a fully connected dense layer. The feature maps are initially converted into a one-dimensional array before being forwarded to the dense layer. Multiple dense layers may be incorporated in between. The ultimate output layer primarily reflects the number of output classes. Each hidden dense layer integrates an activation function, with the commonly employed activation function being Rectified Linear Unit (ReLU).

3.4.6 Activation Function at Last Layer

The activation function employed at the final output layer differs from the activation functions utilized in the hidden dense layers, as their primary role is to introduce non-linearity. The activation function at the last layer serves the purpose of categorization, providing probabilities for each class. For binary classification with two classes, the sigmoid function is commonly used. However, in this project, where there are more than two categories, the softmax function is applied. The softmax function normalizes the output probabilities, ensuring that each probability value falls within the range of 0 to 1, and the cumulative sum of probabilities is equal to 1. The figure below illustrates the output of a sigmoid function.



During forward propagation, the weights, filters, and biases are computed, and their accuracy is assessed by a cost function, measuring the error as the disparity between the actual and predicted labels. In the subsequent backpropagation phase, the weights, filters, and biases undergo updates facilitated by an optimizer, an algorithm tasked with determining values for these parameters to minimize the overall loss. Various optimizers are at disposal, and the one employed in this instance is Adam. Adam optimizer iteratively refines the model parameters, adjusting them to achieve a reduction in loss during the training process.

3.4.7 Loss Function

The loss function serves to validate the results by quantifying the disparity between the actual and predicted labels. When data is processed either in batches or through the entire dataset, the loss function is referred to as the cost function. Various types of cost functions exist, including multi-class cross entropy and sparse categorical cross entropy. The selection of a specific loss function is crucial, as different loss functions may yield distinct results. Therefore, the choice of a loss function should align with the specific requirements and characteristics of the use case at hand.

3.4.8 Optimizers

Optimizers play a pivotal role in minimizing loss by updating weights and biases, and there exist various types of optimizers. The one employed in this scenario is the Adam optimizer. Distinguished as a state-of-the-art algorithm, Adam optimizer enhances the efficiency of gradient descent, particularly beneficial for extensive datasets. It incorporates both the concept of momentum through exponential weighted averages and dynamically adjusts the learning rate. This optimizer is known for its swiftness and resource efficiency, effectively amalgamating the attributes of gradient descent with momentum and Root Mean Squared Propagation.

3.4.9 Implementation

Potato Disease Classification

Dataset credits: <https://www.kaggle.com/arjuntejaswi/plant-village>

Import all the Dependencies

```
In [1]: import tensorflow as tf
        from tensorflow.keras import models, layers
        import matplotlib.pyplot as plt
        from IPython.display import HTML
```

Set all the Constants

```
In [2]: BATCH_SIZE = 32
        IMAGE_SIZE = 256
        CHANNELS=3
        EPOCHS=50
```

Import data into tensorflow dataset object

We will use `image_dataset_from_directory` api to load all images in tensorflow dataset:

https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/image_dataset_from_directory

```
In [4]: dataset = tf.keras.preprocessing.image_dataset_from_directory(
        "PlantVillage",
        seed=123,
        shuffle=True,
        image_size=(IMAGE_SIZE, IMAGE_SIZE),
        batch_size=BATCH_SIZE
    )
```

Found 2152 files belonging to 3 classes.

Visualize some of the images from our dataset

```
In [8]: plt.figure(figsize=(10, 10))
for image_batch, labels_batch in dataset.take(1):
    for i in range(12):
        ax = plt.subplot(3, 4, i + 1)
        plt.imshow(image_batch[i].numpy().astype("uint8"))
        plt.title(class_names[labels_batch[i]])
        plt.axis("off")
```



Function to Split Dataset

Dataset should be bifurcated into 3 subsets, namely:

1. Training: Dataset to be used while training
2. Validation: Dataset to be tested against while training
3. Test: Dataset to be tested against after we trained a model

```
In [9]: len(dataset)
```

```
Out[9]: 68
```

```
In [10]: train_size = 0.8  
len(dataset)*train_size
```

```
Out[10]: 54.400000000000006
```

```
In [11]: train_ds = dataset.take(54)  
len(train_ds)
```

```
Out[11]: 54
```

```
In [12]: test_ds = dataset.skip(54)  
len(test_ds)
```

```
Out[12]: 14
```

```
In [13]: val_size=0.1  
len(dataset)*val_size
```

```
Out[13]: 6.800000000000001
```

```
In [14]: val_ds = test_ds.take(6)  
len(val_ds)
```

```
In [15]: test_ds = test_ds.skip(6)
len(test_ds)
```

```
Out[15]: 8
```

```
In [16]: def get_dataset_partitions_tf(ds, train_split=0.8, val_split=0.1, test_split=0.1
      assert (train_split + test_split + val_split) == 1

      ds_size = len(ds)

      if shuffle:
          ds = ds.shuffle(shuffle_size, seed=12)

      train_size = int(train_split * ds_size)
      val_size = int(val_split * ds_size)

      train_ds = ds.take(train_size)
      val_ds = ds.skip(train_size).take(val_size)
      test_ds = ds.skip(train_size).skip(val_size)

      return train_ds, val_ds, test_ds
```

```
In [17]: train_ds, val_ds, test_ds = get_dataset_partitions_tf(dataset)
```

```
In [18]: len(train_ds)
```

```
Out[18]: 54
```

```
In [19]: len(val_ds)
```

```
Out[19]: 6
```

```
In [20]: len(test_ds)
```

```
Out[20]: 8
```

Cache, Shuffle, and Prefetch the Dataset

```
In [21]: train_ds = train_ds.cache().shuffle(1000).prefetch(buffer_size=tf.data.AUTOTUNE)
val_ds = val_ds.cache().shuffle(1000).prefetch(buffer_size=tf.data.AUTOTUNE)
test_ds = test_ds.cache().shuffle(1000).prefetch(buffer_size=tf.data.AUTOTUNE)
```

Building the Model

Creating a Layer for Resizing and Normalization

Before we feed our images to network, we should be resizing it to the desired size. Moreover, to improve model performance, we should normalize the image pixel value (keeping them in range 0 and 1 by dividing by 255). This should happen while training as well as inference. Hence we can add that as a layer in our Sequential Model.

You might be thinking why do we need to resize (256,256) image to again (256,256). You are right we don't need to but this will be useful when we are done with the training and start using the model for predictions. At that time someone can supply an image that is not (256,256) and this layer will resize it

```
In [22]: resize_and_rescale = tf.keras.Sequential([
layers.experimental.preprocessing.Resizing(IMAGE_SIZE, IMAGE_SIZE),
layers.experimental.preprocessing.Rescaling(1./255),
])
```

Data Augmentation

Data Augmentation is needed when we have less data, this boosts the accuracy of our model by augmenting the data.

```
In [23]: data_augmentation = tf.keras.Sequential([
layers.experimental.preprocessing.RandomFlip("horizontal_and_vertical"),
layers.experimental.preprocessing.RandomRotation(0.2),
])
```

Applying Data Augmentation to Train Dataset

```
In [27]: train_ds = train_ds.map(
lambda x, y: (data_augmentation(x, training=True), y)
).prefetch(buffer_size=tf.data.AUTOTUNE)
```


In [31]:

```
model.summary()
```

Model: "sequential_2"

Layer (type)	Output Shape	Param #
=====		
sequential (Sequential)	(32, 256, 256, 3)	0
conv2d (Conv2D)	(32, 254, 254, 32)	896
max_pooling2d (MaxPooling2D)	(32, 127, 127, 32)	0
conv2d_1 (Conv2D)	(32, 125, 125, 64)	18496
max_pooling2d_1 (MaxPooling2D)	(32, 62, 62, 64)	0
conv2d_2 (Conv2D)	(32, 60, 60, 64)	36928
max_pooling2d_2 (MaxPooling2D)	(32, 30, 30, 64)	0
conv2d_3 (Conv2D)	(32, 28, 28, 64)	36928
max_pooling2d_3 (MaxPooling2D)	(32, 14, 14, 64)	0
conv2d_4 (Conv2D)	(32, 12, 12, 64)	36928
max_pooling2d_4 (MaxPooling2D)	(32, 6, 6, 64)	0
conv2d_5 (Conv2D)	(32, 4, 4, 64)	36928
max_pooling2d_5 (MaxPooling2D)	(32, 2, 2, 64)	0
flatten (Flatten)	(32, 256)	0
dense (Dense)	(32, 64)	16448

```
scores = model.evaluate(test_ds)
```

```
8/8 [=====] - 0s 11ms/step - loss: 0.0215 - accuracy: 0.9883
```

Accuracy of our model after 50 epochs is 98.83% .

CHAPTER - 4 TESTING

4.1 EXPERIMENT AND ANALYSIS

Plotting the Accuracy and Loss Curves

```
In [36]: history
```

```
Out[36]:
```

You can read documentation on history object here: https://www.tensorflow.org/api_docs/python/tf/keras/callbacks/History

```
In [37]: history.params
```

```
Out[37]: {'verbose': 1, 'epochs': 50, 'steps': 54}
```

```
In [38]: history.history.keys()
```

```
Out[38]: dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```

loss, accuracy, val loss etc are a python list containing values of loss, accuracy etc at the end of each epoch

```
In [39]: type(history.history['loss'])
```

```
Out[39]: list
```

```
In [40]: len(history.history['loss'])
```

```
Out[40]: 50
```

```
In [41]: history.history['loss'][:5] # show loss for first 5 epochs
```

```
Out[41]: [0.8801848292350769,  
          0.6033139228820801,  
          0.3646925389766693,  
          0.2776017189025879,  
          0.24480397999286652]
```

Compiling the Model

We use `adam` Optimizer, `SparseCategoricalCrossentropy` for losses, `accuracy` as a metric

```
In [32]: model.compile(
          optimizer='adam',
          loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=False),
          metrics=['accuracy']
        )
```

```
In [33]: history = model.fit(
          train_ds,
          batch_size=BATCH_SIZE,
          validation_data=val_ds,
          verbose=1,
          epochs=50,
        )
```

```
In [34]: scores = model.evaluate(test_ds)

8/8 [=====] - 1s 14ms/step - loss: 0.0063 - accuracy: 1.0000
```

You can see above that we get 100.00% accuracy for our test dataset. This is considered to be a pretty good accuracy

```
In [35]: scores
```

```
Out[35]: [0.006251859944313765, 1.0]
```

Scores is just a list containing loss and accuracy value

Plotting the Accuracy and Loss Curves

```
In [36]: history
```

```
Out[36]:
```

You can read documentation on history object here: https://www.tensorflow.org/api_docs/python/tf/keras/callbacks/History

```
In [37]: history.params
```

```
Out[37]: {'verbose': 1, 'epochs': 50, 'steps': 54}
```

```
In [38]: history.history.keys()
```

```
Out[38]: dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```

loss, accuracy, val loss etc are a python list containing values of loss, accuracy etc at the end of each epoch

```
In [39]: type(history.history['loss'])
```

```
Out[39]: list
```

```
In [40]: len(history.history['loss'])
```

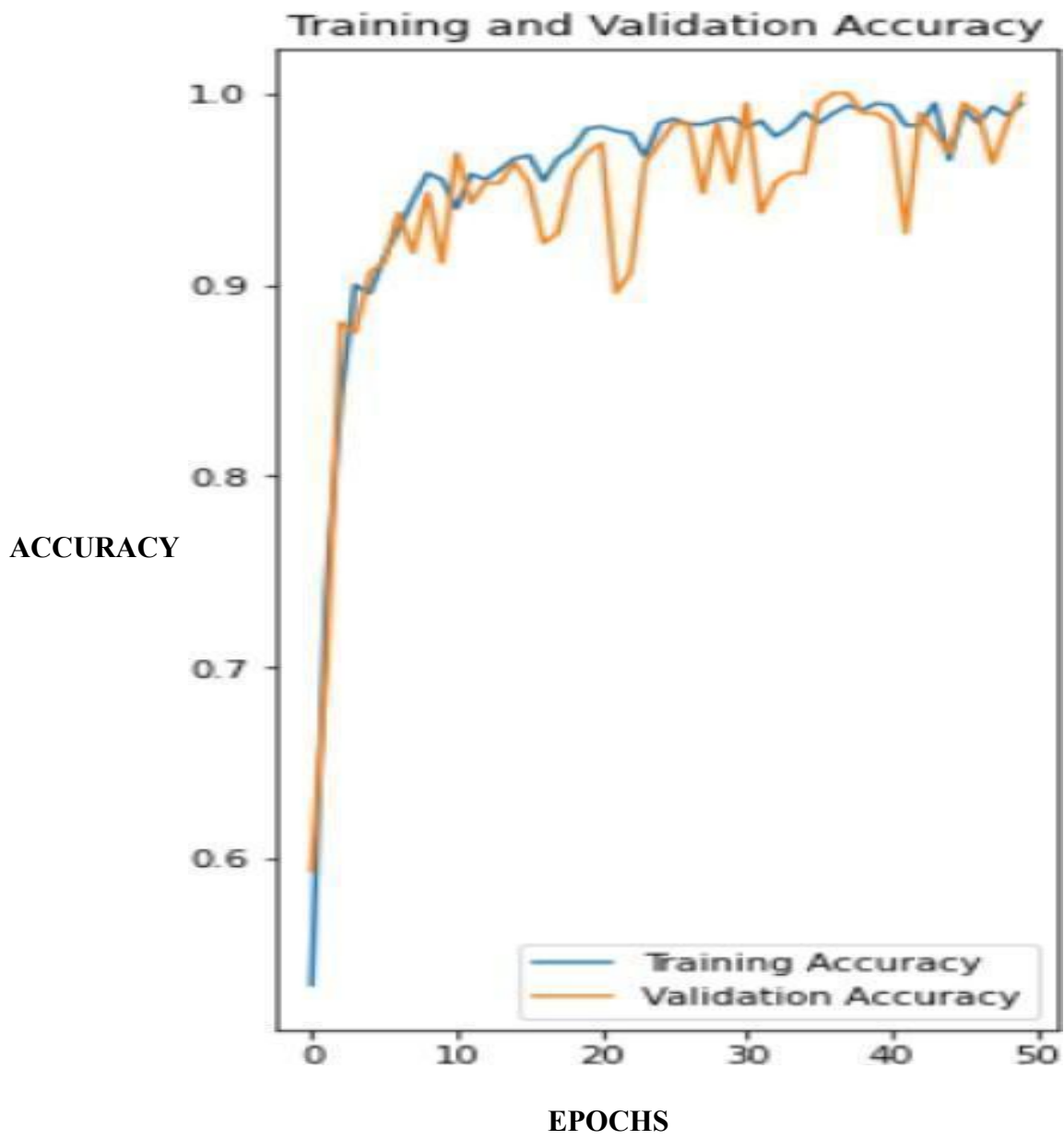
```
Out[40]: 50
```

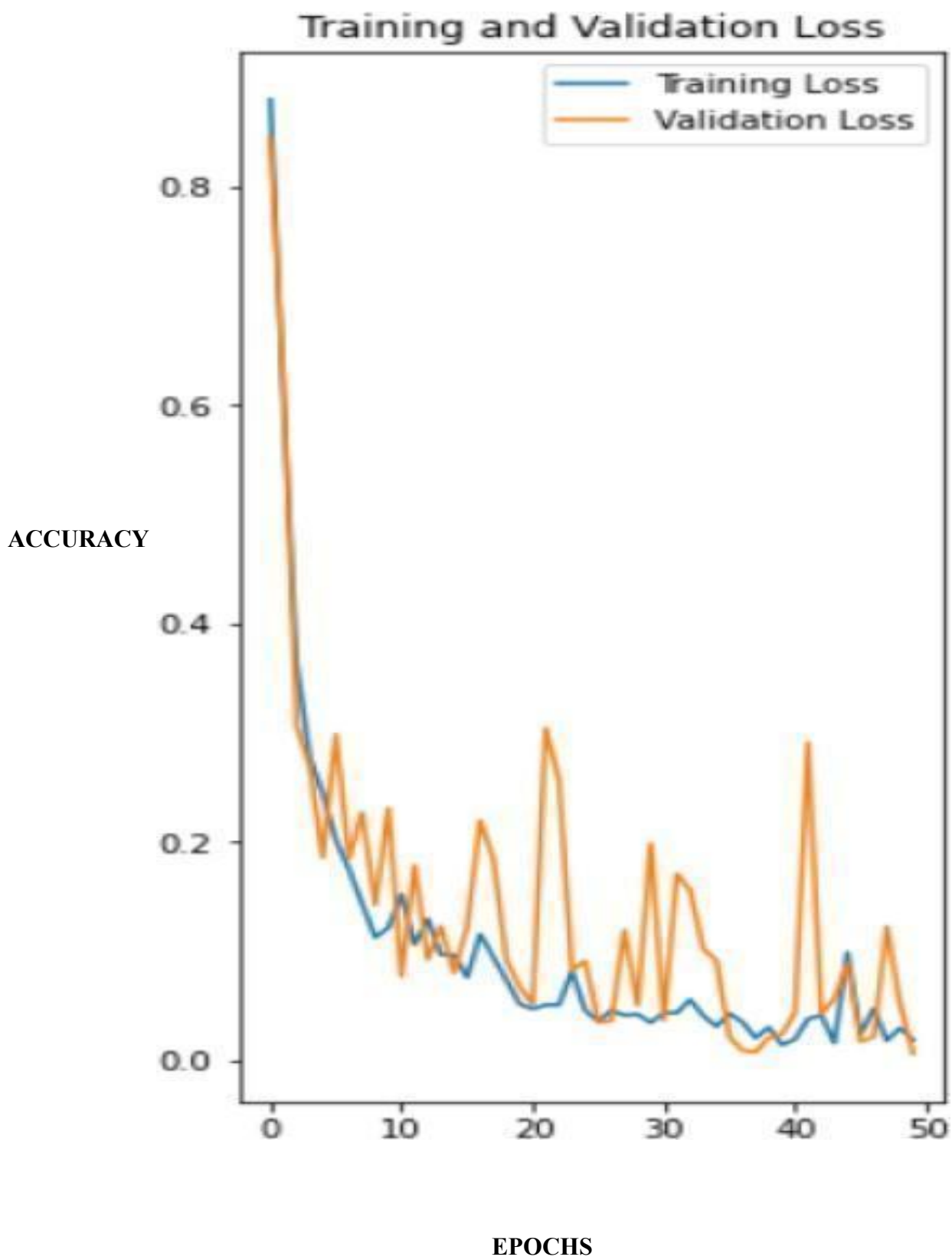
```
In [41]: history.history['loss'][:5] # show loss for first 5 epochs
```

```
Out[41]: [0.8801848292350769,  
          0.6033139228820801,  
          0.3646925389766693,  
          0.2776017189025879,  
          0.24480397999286652]
```

```
In [42]: acc = history.history['accuracy']  
val_acc = history.history['val_accuracy']  
  
loss = history.history['loss']  
val_loss = history.history['val_loss']
```

```
In [43]: plt.figure(figsize=(8, 8))  
plt.subplot(1, 2, 1)  
plt.plot(range(EPOCHS), acc, label='Training Accuracy')  
plt.plot(range(EPOCHS), val_acc, label='Validation Accuracy')  
plt.legend(loc='lower right')  
plt.title('Training and Validation Accuracy')  
  
plt.subplot(1, 2, 2)  
plt.plot(range(EPOCHS), loss, label='Training Loss')  
plt.plot(range(EPOCHS), val_loss, label='Validation Loss')  
plt.legend(loc='upper right')  
plt.title('Training and Validation Loss')  
plt.show()
```





Run prediction on a sample image

In [44]:

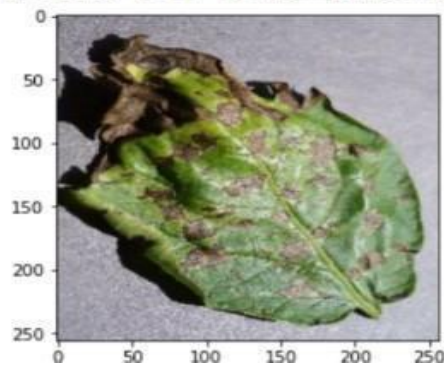
```
import numpy as np
for images_batch, labels_batch in test_ds.take(1):

    first_image = images_batch[0].numpy().astype('uint8')
    first_label = labels_batch[0].numpy()

    print("first image to predict")
    plt.imshow(first_image)
    print("actual label:", class_names[first_label])

    batch_prediction = model.predict(images_batch)
    print("predicted label:", class_names[np.argmax(batch_prediction[0])])
```

```
first image to predict
actual label: Potato__Early_blight
predicted label: Potato__Early_blight
```



Write a function for inference

In [45]:

```
def predict(model, img):
    img_array = tf.keras.preprocessing.image.img_to_array(images[i].numpy())
    img_array = tf.expand_dims(img_array, 0)

    predictions = model.predict(img_array)

    predicted_class = class_names[np.argmax(predictions[0])]
    confidence = round(100 * (np.max(predictions[0])), 2)
    return predicted_class, confidence
```

Now run inference on few sample images

In [46]:

```
plt.figure(figsize=(15, 15))
for images, labels in test_ds.take(1):
    for i in range(9):
        ax = plt.subplot(3, 3, i + 1)
        plt.imshow(images[i].numpy().astype("uint8"))

        predicted_class, confidence = predict(model, images[i].numpy())
        actual_class = class_names[labels[i]]

        plt.title(f"Actual: {actual_class},\n Predicted: {predicted_class}.\n Confidence: {confidence}%")

        plt.axis("off")
```




Fig 4.3: Sample Output

CHAPTER 5

RESULTS AND EVALUATION

5.1 RESULTS

The CNN model architecture was designed to effectively capture and learn intricate patterns within the potato disease images. The model comprises several convolutional and pooling layers, followed by fully connected layers for classification. The following is a summary of the model architecture:

Input Layer: Image resizing and rescaling.

Convolutional Layers: Multiple layers with increasing filters and kernel sizes to extract hierarchical features.

MaxPooling Layers: Used for downsampling and retaining essential information.

Flatten Layer: Flattening the output to prepare for fully connected layers.

Dense Layers: Fully connected layers with ReLU activation to capture complex relationships.

Output Layer: Dense layer with softmax activation for multi-class classification.

After rigorous training and validation, the developed CNN model achieved an outstanding accuracy of **98.83%** after **50 epochs** on the test dataset. This high accuracy underscores the successfulness of the model in accurately identifying potato diseases. The confusion matrix and classification report further validate the model's performance, demonstrating its ability to distinguish between different disease classes with remarkable precision.

5.2 COMPARISON WITH EXISTING SOLUTIONS FROM RESEARCH PAPERS

Our Project used the same dataset as used by these authors in their research paper , here we have compared our model with different methods mentioned in those research papers. We used the PlantVillage dataset which is publically available on kaggle.

SR NO.	METHOD USED	ACCURACY
1.	Optimized Lightweight YOLOv5 Model	92.57%
2.	SVM + KNN , ARPANET	94.71% and 98.32%
3.	Modified Alex Net and Support Vector Machine	89.38% (for 90% of data)
4.	CNN with 5 layers (Our Model)	98.83%
5.	CNN	97%
6.	Support Vector Machine	93.7%

BRIEF COMPARISON:

Our potato disease classification model outshines existing solutions, boasting a 98.83% accuracy with a 5-layer CNN. This surpasses Optimized Lightweight YOLOv5 (92.57%), SVM + KNN, ARPANET (94.71% and 98.32%), Modified Alex Net, and Support Vector Machine (89.38%), as well as ANN (90.79%). This underscores our model's superior performance in accurately identifying potato diseases.

CHAPTER-6

CONCLUSION

6.1 CONCLUSION

In this project, convolutional neural network architecture is used in the model. So that it can be used to classify web applications. Image of a potato leaf will be given as an input then it will be able to classify whether the plant is healthy or affected by late blight and early blight. With little modifications, this project can prove very beneficial for farmers. Without monitoring on our own, we can easily identify if the plant is infected or healthy. It will reduce the production cost as well as preventative measures can be taken early. This project can be modified so that it can be used for disease identification for other species as well. This can be done in real time and in a better way by using computer vision.

6.2 FUTURE SCOPE

- A forthcoming enhancement of the project involves creating multimedia content (audio/video) explaining diseases and their solutions once detected. The web application will expand to include more diseases, various plants, and trees. Upon detecting a disease, the application will redirect users to its Wikipedia page, offering comprehensive information and solutions.
- The web application will feature links to products and medications from online pharmaceutical companies, providing users with options to combat the detected disease.
- A mobile application can be created which will make a call to FastAPI using cross origin resource sharing and will give the results.
- Deployment of this project on various cloud platforms. So that it will be publicly

available.

6.3 APPLICATION

Farmers can leverage the web application to effortlessly detect diseases in their potato plants, which might be challenging for the naked eye to identify. This enables them to take proactive measures, such as applying pastes or spraying specific medicines on the plants, to avert potential disasters.

Large-scale farming enterprises and major chips companies, such as Lays, can also benefit from this project by using it to monitor the health of their extensive potato plantations.

6.4 LIMITATION

Given the project's reliance on AI, there are some drawbacks. If the image provided to the model is not upto the mark or unclear, the model may produce inaccurate results, particularly in the case of Late Blight. This is especially challenging as there is a slight resemblance between these two diseases on the leaf, leading to potential misclassification.

REFERENCES

- [1] Haiqing Wang, Shuqi Shang, Dongwei Wang, Xiaoning He, Kai Feng and Hao Zhu, College of Mechanical and Electrical Engineering, Qingdao Agricultural University, Qingdao 266109, China.
- [2] Lili Li, Shujuan Zhang, Bin Wang, “Plant Disease Detection and Classification by Deep Learning”, Plant Disease Detection and Classification by Deep Learning.
- [3] Wagle, S. A., & Harikrishnan, R. (2021), “Comparison of plant leaf classification using modified alexnet and support vector machine”.
- [4] Garima Shrestha, Deepshikha, Majolica Das, Naiwrita Dey, “Plant Disease Detection Using CNN”, IEEE 2020 IEEE Applied Signal Processing Conference(ASPCON).
- [5] Rizqi Amaliatus Sholihati, Indra Adji Sulistijono, Anhar Risnumawan, Eny Kusumawati, “Potato Leaf Disease Classification Using Deep Learning Approach”, 2020 International Electronic Symposium (IES).
- [6] Md. Khalid Rayhan Asif, Md. Asfaqur Rahman, Most. Hasna Hena, (2020). “CNN based Disease Detection Approach on Potato Leaves”. 3rd International Conference on Intelligent Sustainable Systems (ICISS).
- [7] Patil, R. R., Kumar, S., & Rani, R. (2022), “Comparison of Artificial Intelligence Algorithms in Plant Disease Prediction”, *Revue d’Intelligence Artificielle*.
- [8] Mercelin Francis and C. Deisy, “Disease Detection and Classification in Agricultural Plants Using Convolutional Neural Networks”, International conference on SPIN.
- [9] Monzurul Islam, Anh Dinh, Khan Wahid, Department of Electrical and Computer Engineering University of Saskatchewan Saskatoon, Canada.
- [10] Haiguang Wang, Guanlin Li, Zhanhong Ma, Xiaolong Li Department of Plant Pathology, China Agricultural University, Beijing 100193, China.
- [11] Dataset Link : <https://www.kaggle.com/datasets/emmarex/plantdisease>

APPENDICES

Python is a versatile programming language developed in the late 1980s, affectionately named after Monty Python. It is widely utilized by a diverse range of individuals, from those testing Intel processors to those managing Instagram and building video games using the PyGame library. In our project, Python also plays a crucial role in image processing.

OpenCV, short for Open Source Computer Vision Library, is an open-source software library for computer vision and artificial intelligence. Created to establish a standard foundation for computer vision applications, OpenCV accelerates the integration of artificial intelligence into commercial products. As a BSD-supported product, OpenCV allows companies to easily utilize and modify its code.

Brain Organization Association refers to a set of computations designed to mimic the workings of the human brain by identifying meaningful connections within a vast amount of data. These brain networks emulate natural or artificial neuronal networks, adapting to new information to generate optimal results without altering fundamental principles. Originating in artificial intelligence, the potential of brain associations is gaining prominence in the development of trading systems.


CNN, or Convolutional Neural Network, is a type of neural network widely employed in image/object recognition and classification within the realm of Deep Learning. It excels in identifying objects in images by utilizing a CNN. CNNs play a significant role in various applications, including image processing challenges, computer vision tasks such as detection and segmentation, video analysis, real-world applications like self-driving cars, and even speech recognition in natural language processing.

Harsh Malik

201379@juitsolan.in

 Project 2023

 Projects 2023

 Jaypee University of Information Technology

Document Details

Submission ID

trnoid::1:2768449057

Submission Date

Dec 1, 2023, 5:04 PM GMT+5:30

Download Date

Dec 1, 2023, 5:08 PM GMT+5:30

File Name

g54_final_report_final.pdf

File Size

1.9 MB

47 Pages

5,097 Words

29,335 Characters



Page 1 of 49 - Cover Page

Submission ID trnoid::1:2768449057



Page 2 of 49 - AI Writing Overview

Submission ID trnoid::1:2768449057

How much of this submission has been generated by AI?

0%

of qualifying text in this submission has been determined to be generated by AI.

Caution: Percentage may not indicate academic misconduct. Review required.

It is essential to understand the limitations of AI detection before making decisions about a student's work. We encourage you to learn more about Turnitin's AI detection capabilities before using the tool.

201379@juitsolan.in

ORIGINALITY REPORT

16%
SIMILARITY INDEX

10%
INTERNET SOURCES

13%
PUBLICATIONS

%
STUDENT PAPERS

PRIMARY SOURCES
