Use Object Detection to determine if Crops are Healthy or Diseased based on their Images

1st Prayash Das
dept. of Electrical and Computer Engineering
Stevens Institute of Technology
Hoboken, USA
pdas4@stevens.edu

2nd Prabal Sharma
dept. of Electrical and Computer Engineering
Stevens Institute of Technology
Hoboken, USA
psharma10@stevens.edu

3rd Alex Snyder
dept. of Mechanical Engineering
Stevens Institute of Technology
Hobken, USA
asnyder@stevens.edu

4th Rishab Sharma
dept. of Mechanical Engineering
Stevens Institute of Technology
Hoboken, USA
rsharma10@stevens.edu

I. Introduction

Plant disease detection stands as a crucial task in agriculture, serving as a proactive measure to intervene promptly and prevent the spread of diseases. Timely detection not only protects individual plants but also plays a pivotal role in safeguarding overall crop yield and quality. The overarching goal of our project is to contribute to this imperative task by developing an effective plant disease detection system. To achieve this, we employ various deep learning models, aiming to capitalize on the capabilities of neural networks in discerning subtle patterns indicative of plant diseases. Our dataset is sourced from the PlantVillage dataset, a rich collection of images portraying both healthy and diseased plants across different classes.

A. About the dataset

Our project utilizes a subset of the PlantVillage dataset, focusing specifically on images of potato and bell pepper plant leaves. The leaves were carefully detached from the plants, positioned against either a grey or black background and photographed outdoors using a single digital camera under varying weather conditions, including both sunny and cloudy days. All images in our dataset are standardized to a resolution of 256 x 256 pixels and are in .jpg format. The dataset breakdown is as follows:

Potato:
Potato healthy: 152 files
Potato early blight: 1000 files
Potato late blight: 1000 files

Bell Pepper:
Pepper bell healthy: 1478 files
Pepper bell Bacterial spot: 997 file

II. Related Works

A substantial body of work has been dedicated to plant disease detection, ranging from classical image processing techniques to contemporary deep-learning models. In the domain of plant disease recognition, methods traditionally relied on manual feature extraction, including color co-occurrence matrix, local binary patterns, and texture features, followed by classification using models like support vector machines and decision trees.

However, traditional approaches face limitations in feature extraction, relying on manually designed features and restricting adaptability. Deep learning, with its multilevel network structure, automates feature extraction, allowing for the generation of complex non-linear high-level features. Our project aligns with this evolution, drawing inspiration from successful applications of convolutional neural networks (CNNs) and transfer learning in plant disease detection.

The emergence of deep learning has transformed the field of plant disease identification, empowering models to autonomously discern intricate patterns in data. This advancement, exemplified by convolutional neural networks (CNNs), eliminates the necessity for manual feature engineering and substantially improves accuracy in discriminating between healthy and diseased plants. This project is inspired by three influential articles in the field, namely ”Using Deep Learning for Image-Based Plant Disease Detection” [3] published in the Frontiers journal, and ”Plant Disease Detection with Deep Learning and Feature Extraction using Plant Village” [2] published in the Journal of Computer and Communication, and ’Identification of Plant Disease using Image Processing Technique’ [1] our overarching goal is to build a robust system for early disease detection.

III. Proposed Solution: Model Implementations and Training

In the development of our plant disease detection system, our project employed diverse strategies to address the challenge at hand. These strategies ranged from adopting a conventional approach, where we implemented classification using a Support Vector Machine classifier (SVC), to leveraging fully connected deep neural networks. Subsequently, we delved into the realm of Convolutional Neural Networks (CNNs) and explored the nuances of hyperparameter tuning for these CNNs. Additionally, our exploration extended to the utilization of pre-trained models. This section provides a detailed overview of our model implementations and the subsequent training processes. However, before delving into that, let's examine the steps taken to prepare the data before it could be input into our models.

A. Data Preparation

Data preparation is a crucial step in training effective machine learning models, particularly in image classification tasks. In our project, we utilized TensorFlow, a powerful open-source machine learning library, for its comprehensive set of tools and functionalities designed to facilitate efficient data handling. The following steps outline our approach to data preparation:

1. Loading Data from the Directory: We leveraged TensorFlow's imagedatasetfromdirectory function to seamlessly load images from the "PlantVillage" directory. This function efficiently organizes the images into batches, resizes them to a uniform size (256 x 256 pixels in our case), and shuffles the dataset for better model training.

2. Creating Partitions of the Dataset: The dataset was split into training, validation, and testing sets using the getdatasetpartitionstf function. This function allows us to specify the proportions for each split while ensuring a randomized distribution. Having distinct partitions facilitates robust model training, validation, and evaluation.

3. Optimizing the Dataset: TensorFlow provides efficient methods to optimize datasets for performance during training. By caching data in memory, shuffling elements, and prefetching them with a dynamically adjusted buffer size, we streamlined the input pipeline. This optimization enhances the overall training efficiency by minimizing the time spent waiting for data during the training process.

4. Image Resizing and Rescaling: A critical preprocessing step involves standardizing the size of images. The resizeandrescale sequential layer in TensorFlow performs both resizing and rescaling, ensuring uniformity across all images. Rescaling is essential to bring pixel values into a normalized range, facilitating smoother convergence during model training.

5. Data Augmentation: Data augmentation is an essential technique to enhance model generalization by introducing variations in the training dataset. TensorFlow's data augmentation_sequential layer applies random flips and rotations to the images, diversifying the dataset without the need for additional labeled examples. This augmentation aids in mitigating overfitting and improving the model's ability to handle diverse real-world scenarios.

B. Support Vector Machine Classifier

In our implementation of Support Vector Machine (SVM) for potato disease classification, we embarked on a series of preprocessing steps and model training.The SVM method was approached slightly different than the other methods mentioned in this report, in that scikit-learn was used instead of TensorFlow. The original dataset, comprising images of potato plant leaves, was subject to resizing, reducing the images from 256 by 256 pixels to a compact 20 by 20 pixels. This step, although causing information loss, aimed to simplify the computational demands and assess the model's ability to discern disease patterns in smaller images.

For the SVM model, the dataset was split into training and testing sets, with 80% dedicated to training and 20% to testing. Grid search with cross-validation was employed for hyperparameter tuning, considering different combinations of gamma and C values. The chosen SVM classifier, optimized through grid search, achieved a remarkable test accuracy. This high accuracy, despite the drastic reduction in image size, suggests that the SVM successfully captured distinctive features and patterns associated with potato diseases. The misclassified images were visually inspected, revealing cases where the model struggled to differentiate between classes.

While the SVM demonstrated notable success in potato disease classification, the choice of a 20 by 20 image size raises questions about potential information loss and the robustness of the model across diverse datasets. The experiment provides valuable insights into the adaptability of SVM to reduced image dimensions, and further exploration with larger images and alternative models may shed light on the trade-offs between resolution, computational efficiency, and classification accuracy in the context of potato disease detection.

C. Fully Connected Neural Network Model

In our plant disease detection project, we initiated the model implementation with a fully connected neural network (NN_model) using TensorFlow's Keras API. The architecture consists of preprocessing layers, including resizing and rescaling, to standardize the input images. Additionally, data augmentation layers were incorporated to introduce variability into the training dataset, enhancing the model's generalization. The subsequent dense layers, with ReLU activation functions, serve as feature extractors, progressively learning hierarchical representations. The final dense layer, employing softmax activation, produces probabilities for each class, aligning with the multiclass classification nature of our plant disease detection task. The model was trained for 50 epochs, validated, and evaluated on separate datasets to assess its performance.

We selected a basic fully connected neural network to establish a baseline for plant disease detection. This uncomplicated architecture allows us to gauge dataset characteristics and assess the impact of future model enhancements. Our preprocessing, involving resizing, rescaling, and data augmentation, adheres to image classification best practices for robust learning. The model's simplicity aids interpretation and acts as a starting point for advanced approaches. TensorFlow's Keras API streamlines implementation, providing a high-level interface for efficient neural network development. This method paves the way for further experimentation with alternative architectures and hyperparameter tuning, advancing our pursuit of an accurate plant disease detection system.

D. Convolutional Neural Network Model

In the next phase of our plant disease detection project, we transitioned to a Convolutional Neural Network (CNN) architecture, leveraging the power of deep learning for enhanced feature extraction from images. The model, implemented using TensorFlow's Keras API, consists of multiple convolutional and max-pooling layers designed to capture hierarchical features crucial for image classification tasks.

We thoroughly explored various CNN architectures, experimenting with different configurations to optimize the model's performance. After careful evaluation, we decided to adopt the architecture outlined in the provided code snippet. This architecture begins with preprocessing layers, including resizing and rescaling, followed by data augmentation to introduce variability into the training dataset, promoting robust learning. The subsequent convolutional layers, activated by rectified linear units (ReLU), systematically extract features from the input images. Max-pooling layers then downsample the spatial dimensions, retaining the most relevant information. The flattened layer facilitates the transition from convolutional layers to densely connected layers, introducing non-linearity through activation functions.

The model is trained using the Adam optimizer, a widely used algorithm for optimizing neural networks, with sparse categorical crossentropy as the loss function. During training, we observed the convergence of the model over 20 epochs, assessing its performance on both training and validation datasets.

The chosen CNN architecture offers improved performance compared to the fully connected neural network, especially in tasks involving image data. Its ability to automatically learn hierarchical features makes it well-suited for plant disease detection, where visual patterns are crucial for accurate classification. The incorporation of data augmentation also contributes to the model's robustness and generalization to unseen data, addressing potential overfitting concerns. Overall, this approach marks a crucial step in our pursuit of a sophisticated and accurate plant disease detection system.

Hyperparameter Tuning:
We also delved into hyperparameter tuning for Convolutional Neural Networks (CNNs). This exploration involved two distinct methodologies: a parameter grid search and the advanced KerasTuner library.

1. Parameter Grid Search: We initiated the process by defining a parameter grid encompassing learning rates and the number of layers. Iterating through the grid, we constructed CNN models with varying configurations. Each model underwent training on the provided dataset, and subsequent evaluations on the test set provided insights into their performance. We meticulously recorded the hyperparameters that led to improved accuracy. The training histories, depicted through accuracy curves, offer a visual representation of the learning process for different hyperparameter combinations.

2. KerasTuner: Transitioning to a more automated approach, we leveraged KerasTuner to perform hyperparameter tuning seamlessly. The search space defined for the tuner included the number of layers, number of filters, and the number of dense units. KerasTuner efficiently explored this space, identifying optimal hyperparameters that contributed to enhanced model accuracy. The best configuration was then employed to build a CNN model, which underwent further training and validation. The resulting model's training and validation accuracy curves signify the refined performance achieved through the automated hyperparameter tuning process.

Overall Insights: Due to resource limitations, a comprehensive exploration of hyperparameter tuning for Convolutional Neural Networks (CNNs) was challenging, prompting us to prioritize specific configurations and streamline the process. Despite the constraints, our venture into hyperparameter tuning provided valuable insights into its workings and impacts on our model. As computational resources become more accessible in the future, we aim to conduct a more extensive investigation, exploring a broader range of hyperparameter values and utilizing advanced optimization techniques. Cloud-based solutions or high-performance computing environments will enable a deeper exploration, uncovering nuances for further model improvement.

E. Transfer Learning with MobileNetV2

In the next phase of our project, we embraced the power of transfer learning by incorporating the MobileNetV2 pre-trained model. Transfer learning is a technique that leverages knowledge gained from training a model on a large dataset, often ImageNet in the case of MobileNetV2, and applies that knowledge to a specific task. By utilizing the pre-trained weights of MobileNetV2, which has demonstrated proficiency in recognizing a vast array of features in images, we aimed to enhance our plant disease detection system. This approach allows us to benefit from the wealth of information MobileNetV2 has learned from diverse images, providing a robust foundation for our model to understand and classify features relevant to our specific dataset.

The implementation involves initializing the MobileNetV2 base model with appropriate input dimensions and excluding the top classification layer. By freezing the pre-trained weights (setting trainable=False), we retained the knowledge captured during ImageNet training. We then constructed our custom model by appending a global average pooling layer, followed by dense layers for feature aggregation and classification. The model was compiled using the Adam optimizer, sparse categorical crossentropy loss, and accuracy as the evaluation metric. Training the model involved fitting it to our pre-processed training dataset for 20 epochs with validation data. The performance on the test dataset was evaluated, and accuracy curves were visualized to assess the training progression.

This approach offers a robust and time-efficient solution, capitalizing on the transfer learning capabilities of MobileNetV2. The model's ability to understand complex patterns in plant images contributes to its effectiveness in disease detection tasks. The concise yet informative structure of MobileNetV2 aligns with our goal of achieving accurate results while optimizing computational resources.

F. Exploring Pre-Trained Models for Bell Paper Plant Disease Classification

In this phase of our project, we shifted our focus to classifying Bell Pepper plant data using various pre-trained models, aiming to compare their performance. The Bell Pepper dataset was prepared, and we visually inspected a subset to familiarize ourselves with the data distribution. Subsequently, we partitioned the dataset into training, validation, and testing sets, optimizing their efficiency through caching and prefetching.

To ensure a robust evaluation, we employed three distinguished pre-trained models: ResNet50, EfficientNetB0, and MobileNetV2 : -

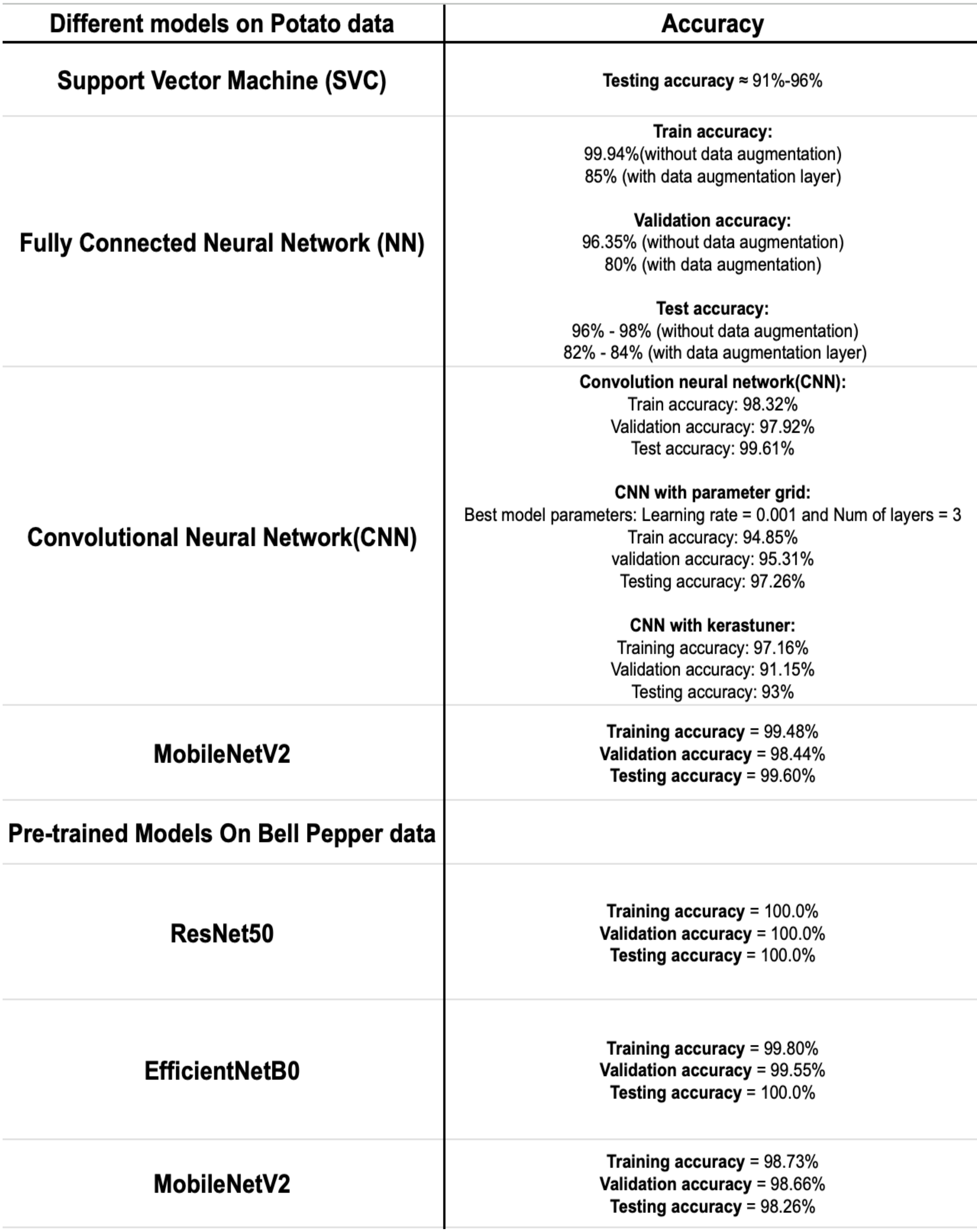
1. ResNet50:
ResNet50, short for Residual Network with 50 layers, is a deep convolutional neural network renowned for its ingenious residual learning architecture. Introduced to address the challenge of vanishing gradients in deep networks, ResNet50 incorporates skip connections, enabling the direct flow of information across layers. This design facilitates the training of remarkably deep networks, making it a popular choice for image classification tasks. ResNet50 has demonstrated superior performance in various computer vision applications, showcasing its ability to capture intricate features in images.

2. EfficientNetB0:
EfficientNetB0 belongs to the EfficientNet family, designed to optimize the trade-off between model size and performance. Introduced by Google, EfficientNet models are characterized by compound scaling, where the network's depth, width, and resolution are systematically increased to find the optimal balance. EfficientNetB0, the baseline model, provides impressive accuracy with significantly fewer parameters compared to other architectures. Its efficiency makes it well-suited for resource-constrained environments, making it a popular choice for a variety of image-related tasks.

3. MobileNetV2:
MobileNetV2, developed with a focus on mobile and edge devices, emphasizes lightweight and efficient neural network architectures. A successor to MobileNetV1, MobileNetV2 introduces inverted residuals and linear bottlenecks, enhancing model accuracy and performance. Its architecture is tailored to minimize computational cost while maintaining competitive accuracy, making it particularly suitable for real-time applications on devices with limited computational resources. MobileNetV2 has found applications in object detection, image classification, and other scenarios where computational efficiency is crucial.

IV. Comparison: Model Performance and Analysis

Implementation of Support Vector Machine algorithm on potato plant disease classification yielded an accuracy of 94.89% on the test dataset which constitutes 20% of the entire dataset. The advantages of implementing this algorithm is it is very straightforward, yields high accuracy and easy to implement. In contrast the computation cost is high, and in the image preprocessing step, during image reduction, many of the information is lost which can affect the performance of SVM.
Implementation of Fully Connected Neural Network on potato plant data yielded an accuracy of 96% to 98% prior to data augmentation and an accuracy of 82% to 84% post augmentation layer after being trained on 50 epochs. This type of Neural Networks offers flexibility and can capture complex relationships in data. However, they may be prone to overfitting while dealing with limited datasets.
The implementation of general Convolution Neural Network (CNN) on potato dataset yielded a test accuracy of 99.61% embedded with hierarchial feature extraction, parameter sharing and spatial hierarchies. They are computationally intensive, requiring substantial resources and are susceptible to overfitting.
Implementation of Pre-trained Models on Bell Pepper data such as ResNet50, EfficientNetB0 and MobileV2 resulted in a test accuracy of 100%, 100% and 98.26% respectively on an epoch of 10.They have impressive accuracy with fewer parameters, making them efficient and they can learn robust features leading to high, training, validation and testing accuracies. However, they are computationally intensive,may not be able to capture intricate patterns unlike deeper architectures.



Different models on Potato data	Accuracy
Support Vector Machine (SVC)	**Testing accuracy** ≈ 91%-96%
Fully Connected Neural Network (NN)	**Train accuracy:** 99.94%(without data augmentation) 85% (with data augmentation layer) **Validation accuracy:** 96.35% (without data augmentation) 80% (with data augmentation) **Test accuracy:** 96% - 98% (without data augmentation) 82% - 84% (with data augmentation layer)
Convolutional Neural Network(CNN)	**Convolution neural network(CNN):** Train accuracy: 98.32% Validation accuracy: 97.92% Test accuracy: 99.61% **CNN with parameter grid:** Best model parameters: Learning rate = 0.001 and Num of layers = 3 Train accuracy: 94.85% validation accuracy: 95.31% Testing accuracy: 97.26% **CNN with kerastuner:** Training accuracy: 97.16% Validation accuracy: 91.15% Testing accuracy: 93%
MobileNetV2	**Training accuracy** = 99.48% **Validation accuracy** = 98.44% **Testing accuracy** = 99.60%
Pre-trained Models On Bell Pepper data	
ResNet50	**Training accuracy** = 100.0% **Validation accuracy** = 100.0% **Testing accuracy** = 100.0%
EfficientNetB0	**Training accuracy** = 99.80% **Validation accuracy** = 99.55% **Testing accuracy** = 100.0%
MobileNetV2	**Training accuracy** = 98.73% **Validation accuracy** = 98.66% **Testing accuracy** = 98.26%

Fig. 1. Comparative Predictive Analysis of Various Models

V. Conclusion and Future Work

In conclusion, our model successfully achieves its objective of predicting diseases in potato plants and bell pepper plants. The success of our model highlights the pivotal role of technology in revolutionizing traditional farming practices. By providing early disease predictions, we empower farmers to make informed decisions, take timely precautions and enhance their crop yields. As we celebrate these achievements, there is an ongoing need for collaborative efforts to further refine and implement such technological solutions in agriculture. Embracing technology in agriculture is not just about solving immediate challenges but paving the way for a sustainable and resilient agricultural future.

References

[1] Abirami Devaraj, Karunya Rathan, Sarvepalli Jaahnavi, and K Indira. Identification of plant disease using image processing technique. In *2019 International Conference on Communication and Signal Processing (ICCSP)*, pages 0749–0753, 2019.

[2] Faye Mohameth, Chen Bingcai, and Kane Amath Sada. Plant disease detection with deep learning and feature extraction using plant village. *Journal of Computer and Communications*, 8(6):10–22, 2020.

[3] Sharada P. Mohanty, David P. Hughes, and Marcel Salathé. Using deep learning for image-based plant disease detection. *Frontiers in Plant Science*, 7, 2016.