# Deep Learning for Early Detection of Plant Diseases in Agriculture

**Sasank Kuppili[1], Meenesh Solanki[2], Avani Vaishnav[3]**
New York University[1,2,3]
sk10404@nyu.edu, mms9905@nyu.edu, av3141@nyu.edu
github.com/early_detection_of_plant_disease

## Abstract

The prompt identification and classification of plant diseases are vital for maintaining agricultural productivity and ensuring food security. Traditional methods, while effective, often require expert knowledge and are time-consuming and labor-intensive. This project seeks to leverage deep learning, specifically Convolutional Neural Networks (CNNs), to develop an efficient model for early detection of common plant diseases. Utilizing the comprehensive PlantVillage dataset, comprising over 50,000 images of healthy and diseased plant leaves across 38 distinct disease classes, our model aims to achieve broad disease coverage and high accuracy. The project evaluates the performance of several CNN architectures, including GoogLeNet, VGG, and ResNet, and experiments with techniques such as transfer learning. The GoogLeNet and VGG architecture perform exceptionally well on the dataset and achieve accuracies of 95.30 % and 93.74% respectively. This project presents a significant step towards using AI in agriculture, providing a potent tool for farmers and agronomists that could contribute to sustainable farming practices.

## Introduction

The rapid and accurate detection of plant diseases plays a critical role in maintaining agricultural productivity and preventing devastating losses in crop yield. Farmers, researchers, and agricultural experts have traditionally relied on manual inspection and laboratory testing to diagnose and manage plant diseases. However, these methods can be time-consuming, labor-intensive, and often require expert knowledge, which may not be readily available in many farming communities around the world. This underlines the need for a more efficient and accessible solution for plant disease detection and management.

In recent years, the advent of artificial intelligence (AI) and, more specifically, deep learning techniques have revolutionized numerous fields, including agriculture. CNNs have the capability to analyze visual imagery and have been extensively applied in image recognition tasks.

The models under consideration for this project include GoogLeNet, VGG, and ResNet. We evaluate these models' effectiveness and performance, and experiment with additional techniques like transfer learning, which enhances our model's accuracy.

This project not only contributes to the existing body of research on AI-based plant disease detection but also potentially paves the way for real-world applications that could benefit farmers and agricultural experts by providing a powerful, accessible tool for early disease detection and management, ultimately contributing to sustainable agricultural practices.

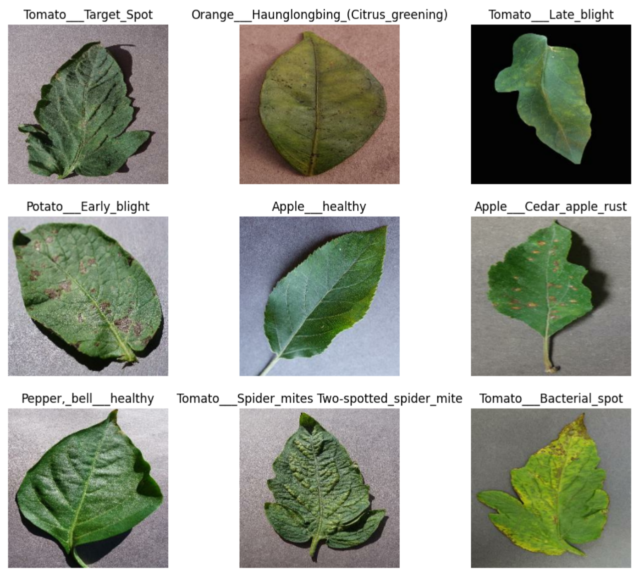

Figure 1: Example of Leaf Imaged from the PlantVillage Dataset

## Literature Review

In their paper, Sharada Mohanty, David Hughes, and Marcel Salathe (2016) explore the application of deep learning techniques for image-based plant disease detection. They use a large dataset of 54,306 images, covering 38 classes of plant diseases and healthy plants. The dataset was divided into training, validation, and test sets in an 80:10:10 ratio. The authors implemented a Convolutional Neural Network (CNN) with 16 layers, including 13 convolutional layers, 2 fully connected layers, and a final softmax layer. They also employed data augmentation techniques to increase the diversity of the training data, which helped improve the model's generalization.

The CNN model was trained using stochastic gradient descent with a learning rate of 0.001 and a mini-batch size of 32. The authors reported an overall accuracy of 99.35% in identifying the correct class. The study demonstrated the potential of deep learning methods for efficient and accurate plant disease detection, paving the way for further research in this area.

This study (Ramcharan et al. 2017) presented a novel approach to detect cassava diseases using deep learning techniques. The authors developed a CNN model to identify and classify four common cassava diseases: Cassava Brown Streak Disease (CBSD), Cassava Mosaic Disease (CMD), Cassava Bacterial Blight (CBB), and Cassava Green Mite (CGM). The dataset used in this study comprised 2,756 labeled images, with 689 images per disease class and an equal number of healthy plant images.

The authors trained the CNN model using the TensorFlow framework, employing a 10-fold cross-validation technique. They also performed data augmentation to increase the diversity of the training dataset. The model achieved a classification accuracy of 88.3% on the test set, highlighting the effectiveness of deep learning techniques in agricultural applications.

In the study done by Konstantinos Ferentinos (2018) explored the use of deep learning techniques for detecting and diagnosing plant diseases. He evaluated various deep learning models, such as convolutional neural networks (CNNs), for their effectiveness in recognizing and categorizing plant diseases from images. The author used a dataset comprising 87,840 images of healthy and diseased plants, covering 14 crop species and 26 disease classes.

The study compared the performance of different CNN architectures, including AlexNet, VGG, GoogLeNet, and ResNet. The author reported that the best-performing model, a modified version of the VGG architecture, achieved a classification accuracy of 99.53%. The study also highlighted the importance of image preprocessing techniques, such as color normalization and data augmentation, for improving the performance of deep learning models in plant disease detection and diagnosis.

## Architecture

**GoogLeNet InceptionV3**: is the third iteration of the original GoogLeNet model, is a convolutional neural network (CNN) designed for image recognition tasks. It is known for its complex but effective architecture that combines the advantages of different types of layers and structures to improve the model's performance.

The architecture of InceptionV3 is organized into a stem and a series of Inception modules, followed by average pooling, dropout, and a fully connected layer.

1. Stem: The stem is the initial part of the network, which performs a few convolution and pooling operations to reduce the spatial dimensions of the input while increasing the depth.
2. Inception modules: The heart of the InceptionV3 model lies in its Inception modules, which are repeated throughout the network. Each Inception module is a small network itself, consisting of several parallel branches that independently perform convolutions and pooling operations before their outputs are concatenated. The key innovation in Inception modules is the introduction of 1x1 convolutions before 3x3 and 5x5 convolutions, which helps to reduce computational complexity while preserving the model's expressive power.
3. Factorizing convolutions: Factorizing convolutions split larger convolutions into smaller ones. For instance, a 5x5 convolution is factorized into two consecutive 3x3 convolutions. This reduces the number of parameters and computational cost without sacrificing the model's ability to capture complex patterns.
4. Asymmetric convolutions: Instead of using square convolution kernels (e.g., 3x3), InceptionV3 employs asymmetric convolutions (e.g., 1x3 followed by 3x1). This allows the model to capture patterns in the data with fewer parameters and computations.
5. Auxiliary classifiers: InceptionV3 also features auxiliary classifiers, which are additional softmax layers inserted in the middle of the network. These classifiers not only contribute to the final prediction but also provide regularization, helping to prevent overfitting during training.
6. Final layers: After the last Inception module, the model applies global average pooling to reduce each feature map to a single value. It then applies dropout for regularization and a fully connected layer for final classification.
7. InceptionV3 is known for its efficiency in terms of computational resources and its high performance in image classification tasks. Its unique architectural features make it suitable for tasks that require the model to capture complex patterns in the data, such as identifying and classifying plant diseases from images.

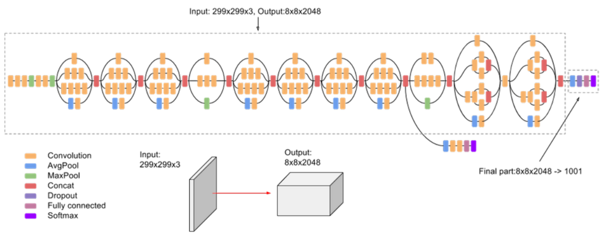

Figure 2: GoogLeNet Architecture

**VGG19 Model**: The VGG19 model, developed by the Visual Geometry Group, (VGG) at the University of Oxford, is a convolutional neural network (CNN) renowned for its simplicity and performance. It is often used in deep learning applications for image recognition and classification.

The VGG19 network is characterized by its depth. It consists of 19 layers with learnable weights: 16 convolutional layers and three fully connected layers. The model also includes five max-pooling layers interspersed among the convolutional layers and a final softmax layer for output.

The VGG19 model follows the architectural style of AlexNet, but with an increased depth. The input to the model is a 224x224 RGB image, and all convolutions use a small 3x3 kernel with a stride of 1 and padding to maintain the spatial dimensions.

After each convolutional layer, the network applies a rectified linear unit (ReLU) activation function. After the final convolutional block, the model uses two fully connected layers, each with 4096 nodes. The final layer uses a softmax activation function to generate probabilities for each class.

Despite its depth, its architecture is straightforward and easy to understand. It is a popular choice for transfer learning because pre-trained weights are readily available, making it an excellent base model for various image classification tasks.

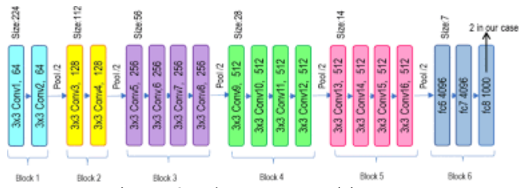

Figure 3: The VGG Architecture

**ResNet50**: a variant of the Residual Network (ResNet) architecture, is a deep convolutional neural network (CNN) known for its ability to alleviate the vanishing gradient problem and achieve high performance in image classification tasks. Developed by Kaiming He and his team, ResNet is characterized by its unique residual connections, which allow the network to efficiently train deeper models.

ResNet-50 contains 50 layers with learnable weights, including convolutional and fully connected layers. The model is organized into a series of residual blocks, each containing several layers. The architecture can be broken down into the following components:

Initial layers: The model starts with a 7x7 convolutional layer with a stride of 2, followed by a max-pooling layer. This helps reduce the spatial dimensions of the input while increasing the depth.

Residual blocks: The core of the ResNet-50 architecture consists of a series of residual blocks. Each block contains several layers, including 1x1, 3x3, and 1x1 convolutions, followed by batch normalization and ReLU activation functions. The unique aspect of these blocks is the residual (or shortcut) connection that skips one or more layers and adds the output of the previous layer to the output of the current layer. This mechanism helps mitigate the vanishing gradient problem and enables more efficient training of deep networks.

Bottleneck layers: ResNet-50 employs bottleneck layers to reduce the number of parameters and computational complexity. These layers use 1x1 convolutions to reduce and then restore the number of channels in the 3x3 convolutions, resulting in a more efficient architecture.

Final layers: After the last residual block, the model applies global average pooling to reduce each feature map to a single value. It then uses a fully connected layer with softmax activation for the final classification.

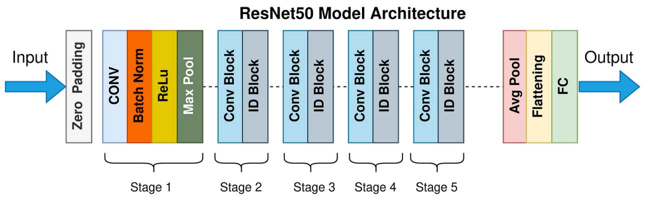

Figure 4: The ResNet-50 Architecture

## Methodology

In the process of developing a robust and efficient model for the PlantVillage dataset, three different models were implemented, each with its unique modifications and strategies.

The following list the variations in the GoogLeNet architecture:

**Model 1:** Transfer Learning with InceptionV3
For our initial model, we utilized the power of transfer learning with the InceptionV3 model. The base model was frozen to preserve the learned features, and a new classification layer was added on top, consisting of a GlobalAveragePooling2D layer, a Dense layer with 256 nodes and 'relu' activation, and a Dense output layer with 38 classes and 'softmax' activation. The model was compiled with 'adam' optimizer,

'categorical_crossentropy' as the loss function, and 'accuracy' as the metric. The model achieved a training accuracy of 98.51% and a validation accuracy of 92.96% by the 20th epoch.

**Model 2:** Fine-tuning InceptionV3
The second model was an extension of the first, with a refined approach. We loaded the best model from the previous version, changed the activation function of the penultimate dense layer to 'sigmoid', and added another dense layer with a 'softmax' activation function. The 'adam' optimizer and 'categorical_crossentropy' were maintained. The model achieved a validation accuracy of 95.30%, showing a significant improvement over the first model.

**Model 3:** Enhanced Robustness and Convergence
The third model focused on enhancing the model's robustness against overfitting and improving its ability to converge to the optimal solution. We introduced a dense layer with 512 neurons and ReLU activation and implemented EarlyStopping callback in addition to ModelCheckpoint and ReduceLROnPlateau. Training stopped at the 13th epoch, and the model achieved a validation accuracy of 95.08%.

**Model 4:** Extended Training with InceptionV3
In the fourth version, we continued training the best model from version 2 for additional epochs. The activation function of the last Dense layer was changed to 'sigmoid', and the number of epochs was increased to 30. Despite achieving a training accuracy of nearly 100%, the validation accuracy fluctuated around 95%, and the validation loss did not show a significant decrease after the 5th epoch.

In summary, these four models represent the evolution of our design strategy for the PlantVillage dataset, each building upon the previous with alterations in architecture, training strategies, and hyperparameters. Despite the varying approaches, all four models successfully leveraged the power of the InceptionV3 model, demonstrating substantial progress towards our goal of a reliable and efficient plant disease classification model.

Table 1: Comparing Training Accuracy of Modified GoogLeNet InceptionV3 Models

| Epochs | Model 1 Accuracy | Model 2 Accuracy | Model 3 Accuracy |
|---|---|---|---|
| 5 | 95.41 % | 98.89 % | 97.57 % |
| 10 | 97.24 % | 99.87 % | 99.96 % |
| 15 | 98.12 % | 99.94 % | Early stopped |

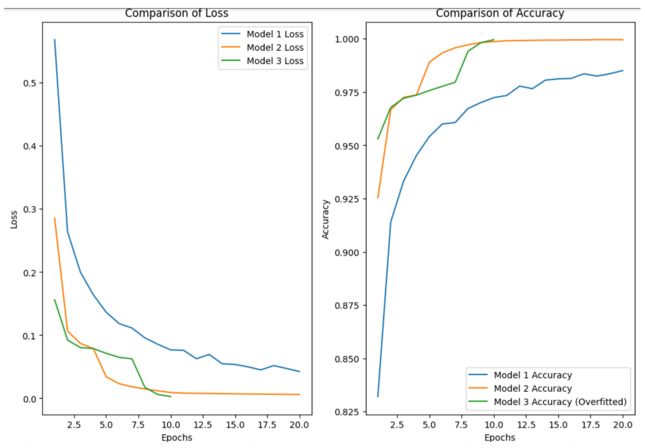

Figure 5: Comparison of Loss and Accuracy of the Inception V3 models

The following are the different modification were made to the VGG19 architecture:

**Model 1:** The first model we experiment with is the VGG19 model. This convolutional neural network model includes 19 layers: 16 convolutional layers, 2 fully connected layers, and an output layer. The model is trained for 20 epochs, achieving a training accuracy of 98.63% and validation accuracy of 93.51% by the end of training. However, the best model was saved at the 6th epoch, when the validation loss was the lowest, indicating that the model started to overfit after this point.

**Model 2:** To improve upon the overfitting issue observed in the first version, changes were made to the model's architecture and training process in the second version of the VGG19 model. The performance on the training data remained high, around 88%, but the validation accuracy was lower at approximately 85%, indicating some degree of overfitting. The best model, as per the validation loss, was saved at the 17th epoch. Suggestions for further improvement included introducing regularization techniques, fine-tuning the model by unfreezing some of the last layers of the base VGG19 model, and experimenting with different architectures or other pre-trained models.

**Model 3:** In the third version of the VGG19 model, further adjustments were made to optimize the model's performance and mitigate overfitting. By the 15th epoch, the model achieved an impressive training accuracy of 97.93%, but the validation loss stopped improving after the 3rd epoch, which suggested overfitting. Some strategies to improve the model's performance were proposed, including implementing early stopping, adding dropout layers, applying regularization, and using data augmentation.

In conclusion, the progression from Version 1 to Version 3 of the VGG19 model demonstrates the iterative nature of deep learning model development. We strived to balance model complexity and computational efficiency while reducing overfitting and improving model generalization on unseen data. Each version of the model provided valuable insights, guiding us towards the final, optimized model and justifying the changes made at each step.

Table 2: Comparison of Training Accuracy of Modified VGG19 Models

| Epochs | Model 1 Accuracy | Model 2 Accuracy | Model 3 Accuracy |
|---|---|---|---|
| 5 | 95.05 % | 79.18 % | 94.93 % |
| 10 | 97.51 % | 84.04 % | 97.0 % |
| 15 | 98.17 % | 86.73 % | 97.93 % |

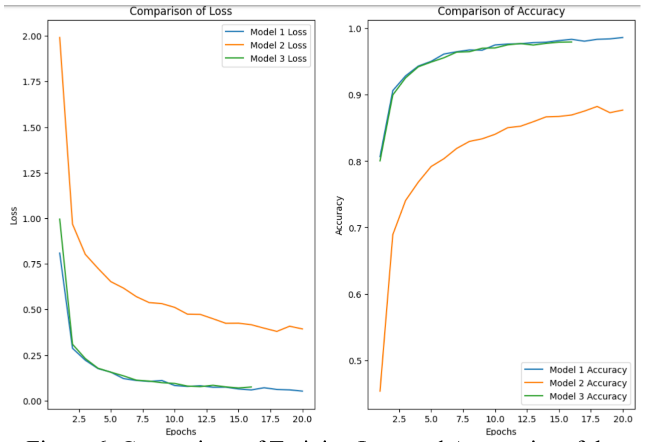

Figure 6: Comparison of Training Loss and Accuracies of the VGG19 models

The following are the variations and modifications we made for the ResNet-50 architecture:

**Model 1**: Transfer Learning
Our base ResNet model consists of 50 layers (48 convolutional layers, one MaxPool layer and one average pool layer.) The base model is frozen and two new layers are added: GlobalAveragePooling and a Dense layer. The output of the last layer is a prediction belonging to one of 38 classes of plant diseases. Softmax is used as an activation function for this model. Early stop is implemented as a callback to the model which allows the model to stop training and prevent overfitting. The batch size for training and validation is fixed at 32. This architecture is trained for 50 epochs and achieves a training accuracy of 57.81% and a validation accuracy of 58.96%.

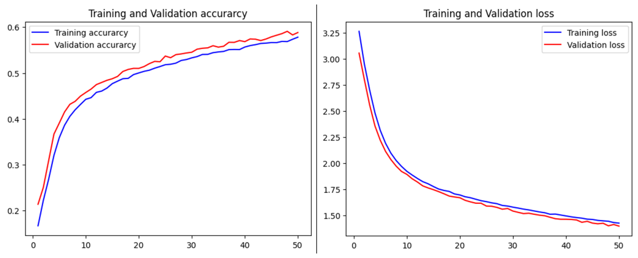

Figure 7: Training and Validation Loss and Accuracy Graphs of ResNet Model 1

**Model 2:** Transfer Learning with Early Stop
In this version of the model, the base architecture is kept the same. In addition to early stopping, reduce on plateau callback is implemented. This allows the model to reduce the learning rate when the validation loss stops improving. The learning rate is updated to 0.00001 when a patience of 3 epochs is reached. In this version of the model the batch size is increased to 128 to allow for faster convergence and is run for the same number of epochs as the previous model. This model achieves a training accuracy of 57% and a validation accuracy of 56.93%.

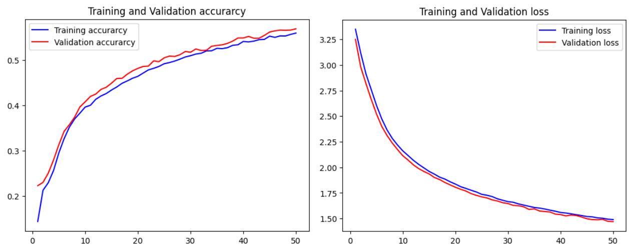

Figure 8: Training and Validation Loss and Accuracy Graphs of ResNet Model 2

**Model 3:** Transfer Learning with Reduce on Plateau
In this iteration of the ResNet architecture, we change the activation function of the last layer to sigmoid to improve the accuracy of the model. Early stop and reduce on plateau callbacks are in place as the previous model. The number of parameters of the last Dense layer is changed from 256 to 512. This model achieves a training accuracy of ___ and a validation accuracy of _____.

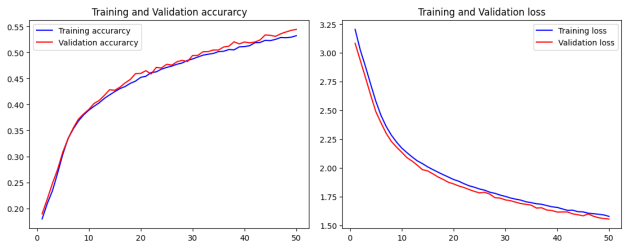

Figure 9: Training and Validation Loss and Accuracy Graphs for ResNet Model 3

Table 3: Comparison of Training Accuracy of Modified ResNet Models

| Epochs | Model 1 Accuracy | Model 2 Accuracy | Model 3 Accuracy |
|---|---|---|---|
| 5 | 35.88% | 29.53% | 94.93 % |
| 10 | 44.27% | 39.66% | 97.0 % |
| 15 | 47.68% | 43.42% | 97.93 % |

## Results

In our exploration of different deep learning architectures for the early detection of plant diseases, we developed and trained three distinct models: GoogLeNet Inception V3, VGG, and ResNet. Upon evaluation, we found that the GoogLeNet Inception V3 model outperformed the other two in terms of accuracy. This finding aligns well with the results documented in the research paper "Using deep learning for image-based plant disease detection" by Mohanty, S. P., Hughes, D. P., & Salathé, M. (2016), which also highlights the effectiveness of the GoogLeNet (V1) architecture in the field of plant disease detection. Our project, thus, not only reinforces the findings of this previous research but also provides a practical implementation of the same, demonstrating the efficacy of GoogLeNet Inception V3 for this specific application.

Table 4: Comparing Models

| **Architecture** | **Best Accuracy** |
|---|---|
| GoogLeNet Inception V3 | 95.05 % |
| VGG19 | 93.51 % |
| ResNet-50 | 58.96 % |

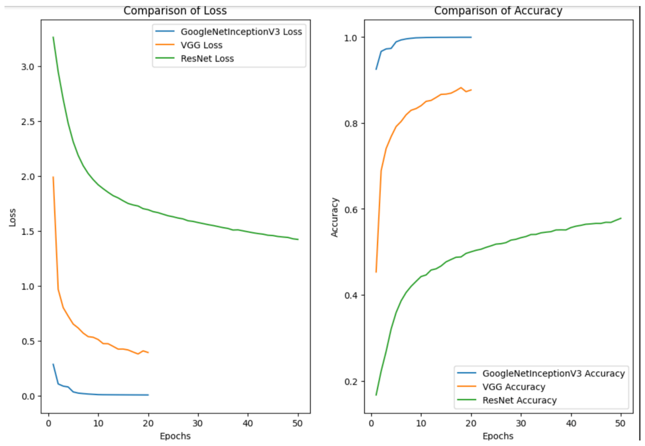

Figure 10: Comparison of Best Versions of the Three Trained Models

## Conclusion

During the initial stages of our project, we planned to incorporate three distinct datasets: the PlantVillage dataset, the Rice Leaf Disease dataset, and the Tomato Leaf Disease dataset. However, we encountered challenges in harmonizing these datasets due to the differing image sizes and inconsistent labeling conventions across them. Despite our efforts to resize the images and reconcile the labels, the integration of all three datasets proved unfeasible.

Consequently, we decided to focus our efforts on the PlantVillage dataset, which provided a substantial and diverse set of images for our models to learn from. We adopted a transfer learning approach, leveraging pre-trained models including GoogLeNet Inception V3, VGG19, and ResNet50, and fine-tuned them on the PlantVillage dataset.

## References

Mohanty, S. P., Hughes, D. P., & Salathé, M. (2016). Using deep learning for image-based plant disease detection. Frontiers in Plant Science, 7, 1419

Ramcharan, A., Baranowski, K., McCloskey, P., Ahmed, B., Legg, J., & Hughes, D. P. (2017). Deep learning for image-based cassava disease detection. Frontiers in Plant Science, 8, 1852

Ferentinos, K. P. (2018). Deep learning models for plant disease detection and diagnosis. Computers and Electronics in Agriculture, 145, 311-318

DATAGEN. ResNet-50: The Basics and a Quick Tutorial.https://datagen.tech/guides/computer-vision/resnet-50/

GOOGLE. Advanced Guide to Inception v3. https://cloud.google.com/tpu/docs/inception-v3-advanced

KERAS. VGG16 and VGG19. https://keras.io/api/applications/vgg/

## Acknowledgements

We thank professor Chinmay Hegde for his guidance, mentorship, and expertise throughout this project. We would also like to thank the staff and facilities at New York University for providing us with the necessary resources to see this project to completion.