Examining Prominent CNN Models Via Application of Image Classification

Sabbir Ahmed, Saad Haider, Sruthy Rajesh Bindhu

Abstract — ResNet, VGG19, and InceptionV3 models have varying levels of success when being applied to certain image types for classification. This study provides a cross-comparison of the three architectures, using images of leaf diseases to test their precision, recall, F1-score, and overall accuracy. These results are compared with a previous finding for a separate research that used images of almonds for image classification [1], for the sake of checking for discrepancies between accuracy levels. Results for leaf disease detection show a high validation accuracy for InceptionV3 and VGG19 at around 98.48% and 95.43% respectively, while ResNet50 struggles at 65.37%. This is in contrast with the almond classification example, which assigns a validation accuracy of 99.65%, 98.95%, and 59.03% respectively. We conclude that for accuracy-related purposes, InceptionV3 is a superior and more consistent model that can produce favorable results, and VGG19 is a fine substitute whose accuracy can vary more, but is still mostly successful at classification tasks. Meanwhile, ResNet50 struggles with multiple image types and has a varied accuracy that appears to be middling at best, and not quite ideal for many scenarios. Limitations include hardware constraints and uniformly 'ideal' images that cannot train the model to detect diseases, for images with lighting conditions or shadows. Further research can be considered in terms of more advanced, recent models and how they fare compared to classical ones. Examinations can also be performed on the computation speed of the aforementioned architectures, along with variant versions of the models and how they perform in comparison to their original counterparts.

I. INTRODUCTION

ResNet, VGG19, and InceptionV3 are some of the most well-known and popular modern Convolutional Neural Network (CNN) models, being fundamental to the application of deep learning techniques to solve image-related problems. These models have their own specialities and strengths, making them stand apart from each other when it comes to image detection. A common feature of these models is that they are designed to enable transfer learning, a system in which a predefined model can be used as a general base that is adapted to a different problem or task, with little sacrifice in terms of accuracy. Researchers often make use of these models for various projects due to the flexibility and utility this feature provides.

It should go without saying that the differences between these architectures make them suitable for different kinds of tasks and purposes. Among the more common tasks is image classification, which has a variety of different applications in the real world. For example, it can allow for easy identification between seemingly similar objects with minute differences, allow for biometrics, and automate certain processes. One project that utilizes this is the "Almond Varieties Image Classification" project by Abhijeet Kaithwas [1], which attempted to distinguish between different types of almonds. The main intent was to create a system that would achieve good accuracy and prove to be reliable and efficient, and for that purpose, three different model architectures were used: ResNet50, VGG19, and InceptionV3. Results show that out of these three architectures, the latter two appear to have done rather well, sitting at a validation accuracy of 99.65% and 98.95% respectively. ResNet50, meanwhile, lags behind at 59.03% accuracy.

The results give a little glimpse into the effectiveness of the models and how they operate, but they belong to just a single example of research work. It is possible that the accuracies found could be vastly different in a different problem set that uses a different type of image. With this in mind, we set out to replicate the system used by the above project and adapt it to a new question: given the same architectures, can we get similar results for a new dataset of images? To this end, we have taken a category of pictures depicting various tomato leaf diseases, originally stored in Kaggle as part of a plant disease detection system [2]. Using this system allows us to see how effective the models' transfer learning capabilities are when applied to a different dataset that is equally as challenging to classify. We then adapted the codebase of the Almond Varieties

Image Classification project and modified it to work with our new set of images, printing out the precision, recall, F1-score, and overall accuracy for each model. The results can be seen in a later section.

II. RELATED WORK

This research directly makes use of a previous project for almond classification [1]. The project was built making use of the three CNN models examined in this paper, ResNet50, VGG19, and InceptionV3. Its purpose was to do the following:

- Automate classification of almond images,
- Achieve high accuracy as much as possible, using the above models to find which works best,
- Create an efficient and scalable system for processing the images,
- Provide a tool that can contribute to agricultural research,
- And assist quality control processes by accurately identifying and sorting almond varieties.

The algorithms made by this project have been adopted for this project, with minor modifications to better support handling images for diseased leaves. The original creator of the above project concluded that InceptionV3 is the best of the models due to its accuracy, with VGG19 being a close second in terms of reliability. ResNet50 did poorly, however, with the inference being made that training issues and data requirements had some effect on the results. Our project is meant to verify this and note down any discrepancies between this project and our own.

III. ARCHITECTURE OVERVIEW

An overview is given on the three CNN architecture models that are used in the experiment this research paper addresses: ResNet50, VGG19, and InceptionV3. Notes on the background of these architectures, the structure of their designs, and their strengths and flaws are discussed.

A. ResNet

Among the first of the modern CNN architectures, ResNet is an architecture developed in 2015 that was designed to solve the vanishing gradient problem that exists in notably deep networks composed of many layers. Prior to ResNet, it was difficult to implement CNNs that were very deep and didn't degrade over time. Vanishing gradients were commonplace for such networks, along with exploding gradients, which made weights grow out of control.

ResNet's solution to the above problems comes in the form of residual blocks, which are composed of several convolutional layers and a skip connection. This connection allows the information obtained in a block to bypass layers, removing the need to learn direct mappings. The network will instead model the difference between the input and the desired output, which reduces the complexity of computation and optimization. The flow of gradients through these connections also acts as a safeguard, ensuring that the overall gradient will remain stable even if gradients directly traveling through the intermediate layers disappear or explode. ReLU activation functions and batch normalization are used to further maintain stability within the layers [3].

Because of this feature, ResNet is responsible for increasing the maximum number of layers allowed in a model, going up to 152 layers. (ResNet50, the model used in this paper, uses 50 layers in its approach.) It also is an effective model that can be used for transfer learning purposes. Though transfer learning, a user can take a pre-trained model and adapt it for a new, different task by keeping its early layers — where general patterns are learned — and removing the later layers used for task-specific training. Due to the flexibility this provides, ResNet is often used for various applications, such as image classification, object recognition, semantic segmentation, and named entity recognition [3].

B. VGG19

VGGNet is an architecture that encompasses a series of CNNs, with popular models including VGG-16 and VGG19. The number refers to the amount of layers utilized in said models [4]. The former model was created in 2014, and the latter in 2016. These models are particularly known for stacking together 3x3 filter grids, which is used to improve the network's abilities in feature extraction and representation learning. All of this is done while keeping parameter efficiency high and steady.

In the specific case of VGG19, it uses sixteen convolutional layers and three fully connected layers to accomplish this task, along with five max-pooling layers and a final softmax layer that is meant for classification and delivering outputs. Stochastic gradient descent and the Adam optimizer are commonly used as optimization algorithms. As for regularization techniques, dropout is used to prevent overfitting, and batch normalization speeds up learning while keeping it stable [5].

VGG19 is heavily used in transfer learning scenarios as a result, being great at computer vision tasks like image classification, object recognition, and feature extraction. It has been notably used for medical applications, and is shown to produce results whose accuracy is comparable, if not superior to other models. However, the efficiency of VGG19 comes at a couple costs. Most notably, it is a rather large model, above 550 MB in size, and requires a lot of time to produce results. Moreover, its great depth and good accuracy comes at the drawback of high computational cost. Interestingly, ResNet was developed to address VGGNet's flaws, notably the vanishing gradient problem, which VGG models suffer from [5].

C. InceptionV3

Belonging to a set of architectures known as the Inception series, InceptionV3 is a model made in 2015 that is designed to perform well on image recognition and classification tasks. The main idea behind this model is to make use of convolution layers known as 'Inception modules' that contain a combination of 1x1, 3x3, and 5x5 convolutional filters, along with pooling layers. The main idea behind these filters and their varying kernel sizes and depth is to capture as many different features as possible from input images [6].

Said features can take the form of edges, textures, shapes, and other similar patterns. Because of the large variety of features that are detected, accuracy is rather high for this model type. Batch normalization and dropout are used in tandem with the various parallel convolutional layers used to achieve such accuracy. They improve its training efficiency and generalization capabilities, respectively [7]. Furthermore, a 'bottleneck' technique is applied before applying large convolutional layers, being used to lower the number of input features sent into the layers. This is used to lower the cost of computation for the model.

Being trained on ImageNet, InceptionV3 is built to have a set of general features already learned in its system, making it very useful for transfer learning scenarios. Medical imaging, agriculture, and quality control are some of the applications that make use of this flexible architecture, taking its pre-existing knowledge and refining it to be applicable to their specific fields of interest [7]. Accuracy is a strong point of InceptionV3, and moreover, it has surprisingly fast inference capabilities, unlike VGG19. This is in spite of the fact that its computational complexity remains rather high, being the main flaw of InceptionV3 models.

IV. DATASET AND PREPROCESSING

The tomato leaf disease dataset used in this project was stored in Kaggle by Tairu Emmaneul, [2] and belongs to a larger set of data compiled by Sharada Mohanty for various plants, the PlantVillage Dataset [8]. The subset stored in Kaggle holds images for peppers, potatoes, and tomatoes that depict their state of health. Due to resource limits, we narrowed this subset further down, focusing on four diseases related to tomato leaves: Two-spotted Spider Mite, Target Spot, Tomato Mosaic Virus, and Tomato Yellow Leaf Curl Virus. These images were cleaned, organized, and split into 5,327 images for training and 1,334 for validation. This eighty-twenty split gave us enough examples to learn features effectively while keeping a solid portion for checking how generalized our models are. The dataset did not have perfectly equal numbers of images for each class, which we handled by generating class weights and incorporating them into the training process. Doing this saved our models from leaning too heavily toward classes that appeared more often. We also made use of many on-the-fly augmentations, such as rotations, width and height shifts, zoom, shear, and flips, in order to avoid overfitting and give the models more visual diversity to learn from. All images were normalized before being fed into the model networks.

V. METHODOLOGY

The "Almond Varieties Image Classification" project's algorithm created models for ResNet, VGG19, and Inception V3 simultaneously. When modifying this algorithm, we split them into three separate algorithms for ease of use and efficiency.

The top fully connected layers were removed for all models, with pre-trained ImageNet weights loaded for each; ImageNet also served as a feature extractor for InceptionV3 (due to its ability to capture features at multiple scales via factorized convolutions). The custom head included a Global Average Pooling layer, a Dense layer of 1024 units with ReLU activation, and a Dropout layer of 0.5 to regularize the network. We employed a fine-tuning strategy where the first 200 layers were frozen.

To test the efficacy of residual learning, we employed ResNet50. The custom head included a Dense layer of 512 units with L2 Regularization of 0.01 to aggressively prevent overfitting, followed by Dropout. Unlike the other models, ResNet50 utilized an EarlyStopping callback to halt training if validation loss plateaued.

We utilized VGG19 as a baseline for its deep, uniform architecture using 3x3 filters. The classification head mirrored the ResNet configuration with 512 units and Dropout. We adopted a specific fine-tuning strategy where layers up to index 12 were frozen, while deeper layers were allowed to update.

All models were optimized using the Adam optimizer, with a learning rate of 0.001 applied to each. We employed Categorical Cross-Entropy as the loss function.

VI. RESULTS

A. InceptionV3 Performance Analysis

In our preliminary results, it was clear InceptionV3 had the highest performance metrics of the study, with a final validation accuracy of 98.48% and a validation loss of 0.0508.

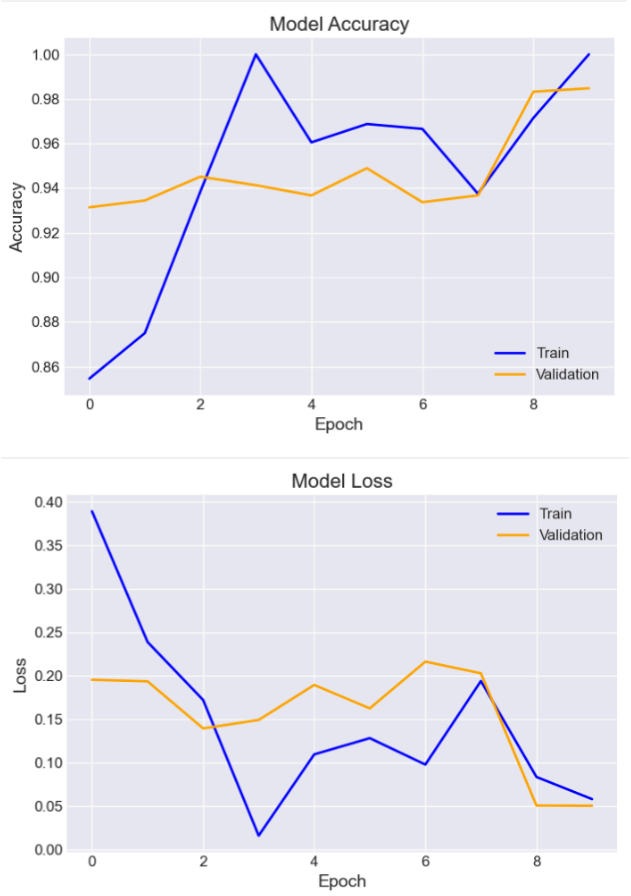

Figure A. Graphs of InceptionV3 accuracy and loss over 10 epochs.

Figure A (top) depicts the model accuracy, with the validation set starting past the 90% threshold from the very first epoch. It eventually has a sharp climb starting from epoch 7. Notably, the training set diverges at certain points of the graph but eventually drops down to converge with the validation line, indicating that the dropout layers and data augmentation successfully prevented overfitting.

Figure A (bottom) is the model loss and shows how validation loss rapidly decayed to below 0.2 within the first epoch and stabilized near 0.05 by the final epoch. The curve is mostly smooth and implies the learning rate of 0.001 was decently optimal for this architecture's convergence rate.

Figure B depicts the confusion matrix for InceptionV3. The heatmap shows a nearly perfect diagonal line, indicating correct predictions. In particular, the model shows perfect matches for the Tomato Mosaic Virus and Target Spot classes, with no false positives or negatives appearing. Spider Mites and Yellow Leaf Curl Virus show a tiny amount of confusion, with the former being falsely predicted as the latter.

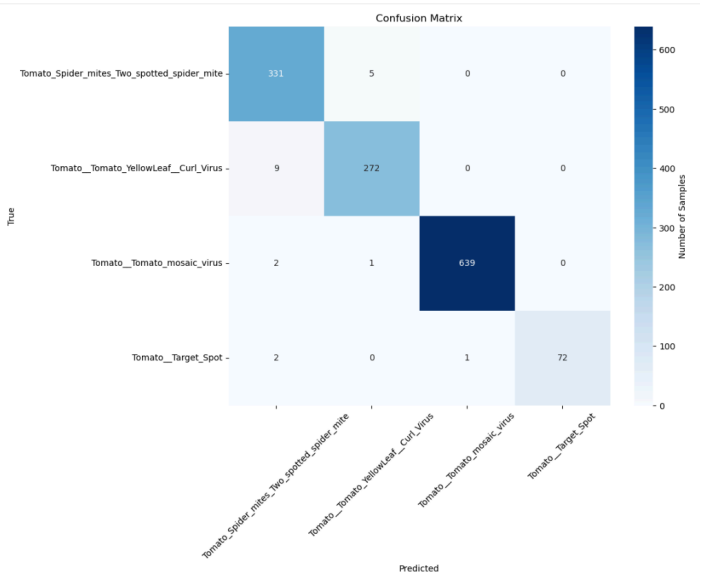

Figure B. Confusion matrix for InceptionV3 classifications.

B. VGG19 Performance Analysis

The VGG19 model is a strong runner-up, holding a final validation accuracy of 95.43% and a validation loss of 0.1413.

Figure C (top) shows how VGG19's accuracy curve has a more gradual learning trajectory compared to InceptionV3. The model started with much lower accuracy in the first epoch but showed consistent improvement. By epoch 4, the validation accuracy spiked past the training accuracy, and both lines stabilized by the final epoch.

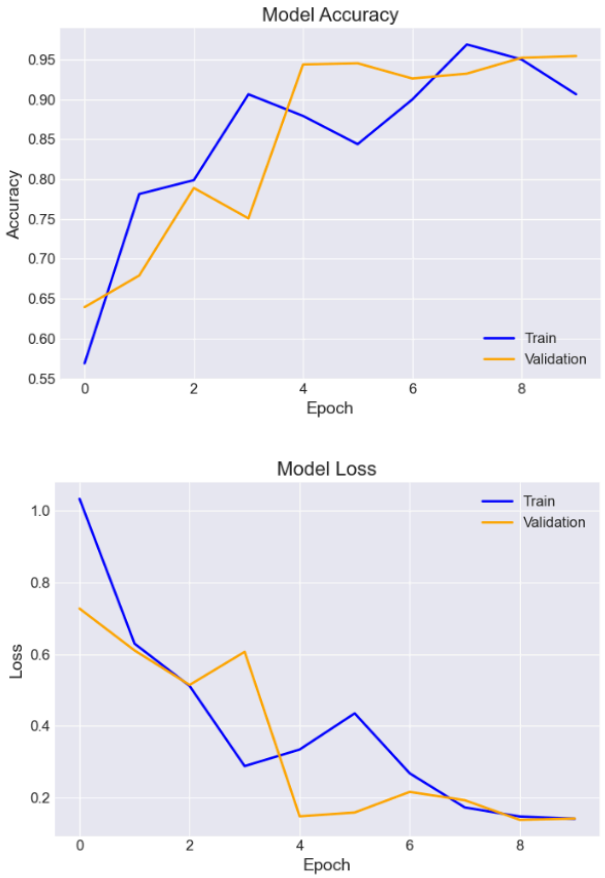

Figure C. Graphs of VGG19 accuracy and loss over 10 epochs.

Loss decreased steadily as seen in Figure C (bottom), but remained higher than InceptionV3 throughout the training process. There is a slight gap between training and validation loss towards the later epochs, hinting at a brief period of mild overfitting that the dropout layer eventually fixed.

The confusion matrix in Figure D results show largely positive results, with the majority of predictions being accurate (as shown by the diagonal line). However, distinct error clusters are visible, with the model struggling more at distinguishing Spider Mites from Target Spot compared to InceptionV3. It also confused Mosaic Virus with Spider Mites. Despite this, the precision for viral classes remained high, which confirms the effectiveness of VGG19 as an image classification model.

C. ResNet Performance Analysis

As it did in the almond classification project, the ResNet50 model underperforms compared to the other architectures, with a final validation accuracy of 65.37% and a high validation loss of 0.9093.

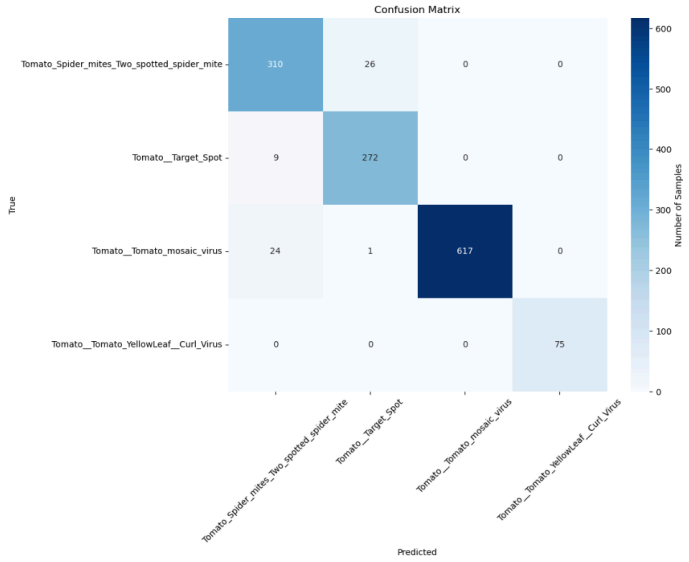

Figure D. Confusion matrix for VGG19 classifications.

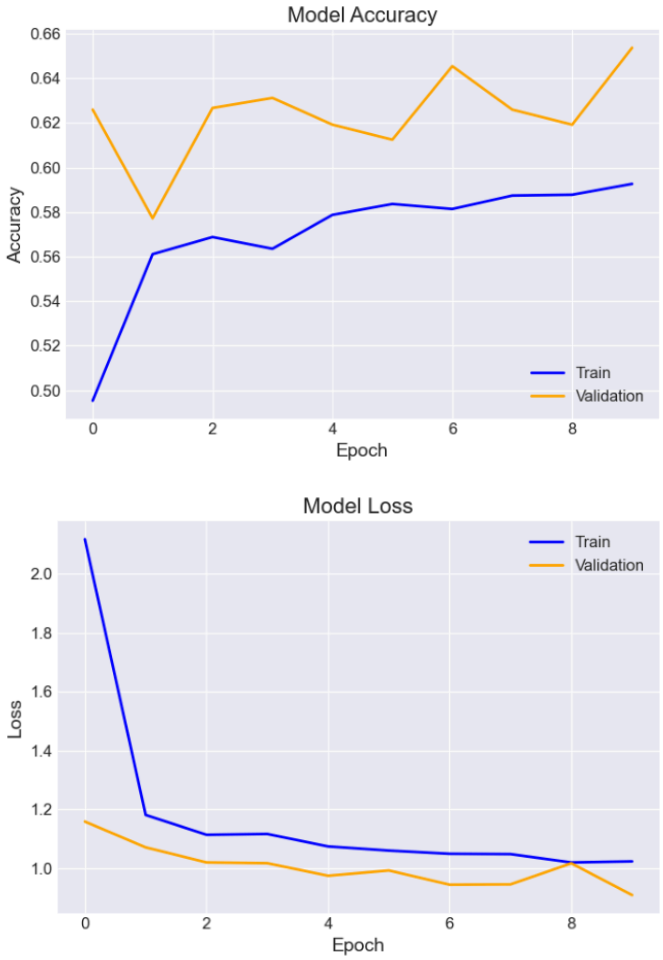

Figure E. Graphs of ResNet50 accuracy and loss over 10 epochs.

The accuracy graph in Figure E (top) displays a volatile and non-converging trajectory. The validation accuracy oscillated around 60% to 65% and failed to show the monotonic increase seen in the other models. This “flatlining” behavior is a sign that the model got stuck in a local minimum early in the training process.

As seen in the loss graph in Figure E (bottom), The loss remains high, hovering around 1.0. Unlike the other models where loss gradually dropped near zero, the ResNet50 loss curve shows almost no improvement after the second epoch, confirming that the weights were not being effectively optimized.

The heatmap for ResNet’s confusion matrix in Figure F reveals a critical failure mode. The column for Tomato Yellow Leaf Curl Virus doesn’t have any correct predictions, meaning the model failed to classify a single image of this disease correctly. Instead, it heavily misclassified these images as Spider Mites or Mosaic Virus. This represents a complete feature collapse for that specific class. In general, false predictions occur frequently in the confusion matrix, with highlights being Target Spot being frequently mistaken for Spider Mites. Mosaic Virus appears to be the disease that the model has best learned how to identify, out of all the classifications, which may indicate how distinguishable it is out of the four classifications.

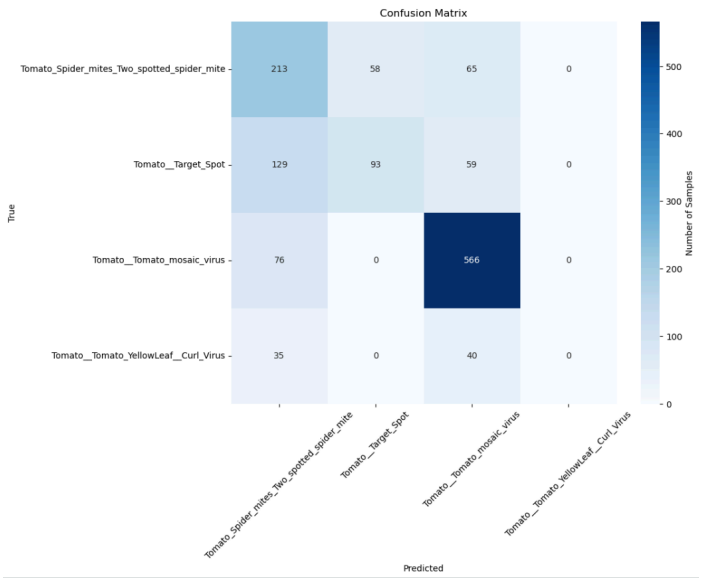

Figure F. Confusion matrix for ResNet50 classifications.

VII. ADDITIONAL FINDINGS

Our analysis yielded three specific findings regarding the application of deep learning to plant disease detection.

First, we found that architecture width appears more beneficial than depth for this specific fine-grained classification task. InceptionV3, which utilizes wide "Inception modules" to capture features at multiple scales simultaneously (using 1x1, 3x3, and 5x5 filters), outperformed the deeper ResNet50 and VGG19 architectures. This suggests that plant disease symptoms manifest at varying scales—from tiny necrotic spots to global leaf curling—and an architecture designed to look at multiple scales at once is better suited to capture these variances than a standard deep stack of filters.

Model	**Val. Accuracy**	**Val. Loss**	**Precision (Weighted)**	**Recall (Weighted)**	**F1-Score (Weighted)**
Inception V3	98.48%	0.0508	0.99	0.99	0.99
VGG19	95.43%	0.1413	0.96	0.96	0.96
ResNet50	65.37%	0.9093	0.62	0.65	0.62

Figure G. Cross-comparison table for accuracy across InceptionV3, VGG19, and ResNet50.

Second, we interpreted the failure of the ResNet50 model as a result of hyperparameter conflict rather than architectural incapacity. While ResNet50 was trained with the same learning rate of 0.001 as the Inception and VGG models, it seemed to be adversely affected by this value. ResNet also had aggressive L2 regularization in its classification head, which appeared to create compounding problems. The learning rate caused the model to oscillate rather than settle into a global minimum, while the regularization likely suppressed feature activation too aggressively given the dataset size. This finding highlights that ResNet architectures are significantly more sensitive to hyperparameter tuning than VGG or Inception architectures in transfer learning scenarios.

Third, we found that the use of computed class weights effectively negated the issue of dataset imbalance. The Tomato Mosaic Virus class had significantly more images than the Target Spot class. Despite this, both InceptionV3 and VGG19 achieved their highest precision scores on these classes. This indicates that mathematically weighting the loss function is a sufficient strategy for agricultural datasets, which are rarely perfectly balanced in real-world collection scenarios.

VIII. LIMITATIONS AND FURTHER RESEARCH

Of the limitations that impacted our experiments, the constraint of computational resources was what impacted our research the most. Training deep Convolutional Neural Networks requires substantial GPU acceleration, while we were

only able to operate with limited hardware resources, which restricted our ability to train on large datasets. We had no access to high-performance cloud computing infrastructure either, which would have allowed for larger batch sizes and more extensive training epochs. This forced us to settle with a relatively small subset of data with only four classes, and even then, we required up to multiple hours of runtime to train on our image set. Using only a subset of the data may also have affected ResNet50, with the lesser amount of data not being sufficient for it to converge, unlike with the more efficient InceptionV3. To overcome these hardware limitations in a future project, it would be ideal to leverage enterprise-grade cloud computing platforms that can perform extensive hyperparameter tuning. This in particular would be great for ResNet50, as this would allow us to provide a more appropriate learning rate for its architecture, alongside the general improvements to efficiency and accuracy results.

Additionally, while the dataset provided high-quality images, they were captured under controlled laboratory conditions with uniform backgrounds. In other words, they were ideal pictures with clear indications of the object they were meant to display, and with no variation in lighting, shadows, or background clutter. The real world is full of such inferences that can throw off a classification model, and the lack of environmental variation to train on is a limiting factor to the applicability of the model. While it is not necessarily our main goal to provide a model that can be used in real-world settings, it would be more beneficial to make use of such factors in order to get a more 'accurate' result for model performance. For future research, it would thus be ideal to prioritize validating these models in such real-world agricultural settings, so that we can get an idea of how these models work in a practical field.

Notwithstanding these limitations, there are several avenues of research that can be done on this project. For one, while accuracy is an important factor for any CNN model, users are also concerned about efficiency and computation speed. It would be worthwhile to expand this project to cover this topic in tandem with accuracy, with the goal being to find a model that holds the "best of both worlds." Metrics would need to be considered on what constitutes such a model and how to rank them accordingly. Furthermore, beyond the original three models that were used for this project, we can also consider adding variants of these models, or even other newer models that have appeared in recent years, to see how well they perform in comparison. For the variants, the idea would be to see where they perform better in comparison to the original model, and as for recent models, it would be useful to chart improvements in the newer technology over the older architectures, if any.

Other ideas include further testing with other image datasets. While this project bears similar results to the almond classification project, the two sets of data points can be false positives for general model performance. Both projects also include agricultural images, which may have skewed the results; using images from vastly different fields can show if any of the models would perform better or worse in different categories.

IX. CONCLUSION

When comparing the results of this paper with the accuracy scores achieved by the almond classification project, we find that the models remained more or less consistent in their effectiveness. Our comparative analysis revealed that the InceptionV3 architecture is the optimal choice for this task, achieving a validation accuracy of 98.48% and demonstrating superior stability compared to VGG19 and ResNet50; this is in comparison to the almond classification project's 99.65% accuracy for the same model, a very similar value that implies that minor variations may be typical for this model's accuracy. VGG19 manages to hold out at 95.43% accuracy, compared to 98.95% accuracy from the other model, which shows how VGG19's model doesn't have the same level of consistency as InceptionV3. Finally, ResNet50 does poorly at 65.37%, though this is an improvement from the 59.03% accuracy achieved by the almond classification project. With both projects showing similar results, this study shows some level of conformity in the models in how they behave, and how accurate they are. That said, there is room to see if ResNet could perform on par with the other models if given a more appropriate learning rate.

Generally, our research shows the ability of classical deep learning models in differentiating between visually similar plant diseases, such as Spider Mites and Target Spot, with a clear bias toward InceptionV3 as an accurate model due to its multi-scale feature extraction abilities. ResNet50 struggles to prove itself as a reliable model, meanwhile, due to its unsteady accuracy, which may be correlated with its sensitivity to the learning rate. It should go without saying, however, that these classical models are not solely defined by their accuracy. They represent different methodological approaches to CNNs and deep learning, with different strengths and weaknesses. Most of all, they represent different steps that were taken to advance technological understandings and capabilities. Their existence is something to learn and iterate upon, and their mistakes a lesson for newer architectures to keep in mind.

X. REFERENCES

[1] A. Kaithwas, “Almond Varieties Image Classification,” Github, Jun. 07, 2024. https://github.com/abhisheks008/DL-Simplified/tree/main/Almond%20Varieties%20Image%20Classification

[2] T. Emmanuel, “PlantVillage Dataset,” *www.kaggle.com*, 2018. https://www.kaggle.com/datasets/emmarex/plantdisease

[3] “Residual Neural Network - an overview | ScienceDirect Topics,” www.sciencedirect.com. https://www.sciencedirect.com/topics/computer-science/residual-neural-network

[4] S. Bangar, “VGG-Net Architecture Explained,” Medium, Jun. 28, 2022. https://medium.com/@siddheshb008/vgg-net-architecture-explained-71179310050f

[5] “VGG-19 Convolutional Neural Network - an overview | ScienceDirect Topics,” www.sciencedirect.com. https://www.sciencedirect.com/topics/computer-science/vgg-19-convolutional-neural-network

[6] N. AI, “InceptionV3,” Medium, Feb. 20, 2023. https://medium.com/@nocodingai/inceptionv3-10e6f48e4553

[7] “What is InceptionV3,” Activeloop.ai, 2023. https://www.activeloop.ai/resources/glossary/inception-v-3/

[8] spMohanty, “GitHub - spMohanty/PlantVillage-Dataset: Dataset of diseased plant leaf images and corresponding labels,” GitHub, 2025. https://github.com/spMohanty/PlantVillage-Dataset/tree/master (accessed Dec. 06, 2025).