Disease Detection in Crops Using a Convolutional Autoencoder and Convolutional Neural Network Hybrid Model

Alexandros Manioudakis (amanioud), Shruti Panse (spanse), Roshan Parikh (roshanparikh), Rio Wombacher (rwombach)

1 Introduction

Plant diseases pose a serious threat to global food security, and early, accurate detection is critical to minimizing crop damage and ensuring healthy agricultural yields. Although deep learning has proven highly effective in image-based disease classification, many current models require substantial computational resources, which makes them impractical for real-world use on farms, where access to powerful hardware can be limited.

Our goal was to develop a model that could accurately detect plant diseases from leaf images while remaining lightweight enough to run efficiently on lower-resource devices, such as smartphones or edge computing units that are deployed in the field. We wanted to find a balance between both accuracy and efficiency so farmers and researchers, and even at-home gardeners could rely on fast, accessible disease diagnostics without the need for cloud-based systems or high-end GPUs.

Our project built off of previous work from researchers Punam Bedi and Pushkar Gole (2021) from the University of Dehli. They proposed a hybrid model combining a Convolutional Autoencoder (CAE) and a Convolutional Neural Network (CNN) to identify bacterial spot disease in peach leaves. Their design achieved high accuracy while keeping the number of trainable parameters relatively low, demonstrating that it was possible to retain strong performance even with a compact architecture.

Based on their architecture, we set out to take on a more complex problem: classifying 33 different plant disease classes using a much larger and more diverse subset of the PlantVillage dataset, which contains over 54,000 images. Our hybrid model used the CAE to compress high-dimensional images data into a smaller latent representation, which was then passed into our CNN for final classification. This design helped both reduce parameter count and generalize better by learning low-level feature representations.

By prioritizing both accuracy and scalability, our project pushed toward real-world solutions in plant disease detection, creating tools that are not just high-performing in a lab but are practical enough to empower farmers, researchers, and home gardeners alike with reliable, low-cost diagnostics that both help support healthier crops and create more resilient food systems.

2 Methodology

2.1 Data

Data was obtained from the PlantVillage dataset. This dataset had 54,303 images of plant leaves of various species and diseases. For example, the dataset included images of healthy peach leaves and peach leaves with bacterial spot disease. In total, there were 14 species included in the dataset, and 38 total species-disease (or healthy) combination classes.

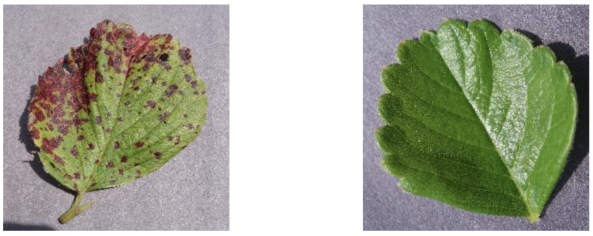

Figure 1: *Example of strawberry leaf with leaf scorch disease (left) and healthy strawberry leaf.*

2.2 Preprocessing

We started out with 38 classes and 54,303 images. These classes were organized as "Plant Name – Disease/Healthy." However, some of the species did not have both a diseased and healthy class, so those files were removed, leaving us with exactly 40,000 images and 33 classes.

The data was already organized into an 80-20 training-validation data split, but we wanted a 70-20-10 training-validation-testing split. We used the `ImageDataGenerator` class from The Tensorflow Keras library to preprocess the images into tensors and to split off 12.5% of the data from the training set to form the testing set, creating a 70-20-10 split. Images were resized to have a resolution of 256×256 pixels. To increase the robustness of our training process, we also performed random vertical flips, rotations (up to 20^{o}), and vertical and horizontal shifts (up to 10%) of the images.

Two types of data labels were created. The first type was for binary classification, where labels represented healthy or unhealthy plants. The second was for multiclass classification, with all remaining 33 class labels. Since testing labels were one-hot-encoded, binary labels had length 2 (example: $[0, 1]$), and multiclass labels had length 33 (example: $[0, \ldots, 1, \ldots, 0]$).

2.3 Architecture

The preprocessed images were fed into a Convolutional Autoencoder (CAE), which compressed them into lower-dimensional latent representations. This compression allowed the model to retain only the most important features necessary for identifying plant diseases while discarding redundant information.

The compressed features extracted from the CAE's bottleneck layer were subsequently passed to a Convolutional Neural Network (CNN) classifier. This CNN produced either binary predictions, classifying each plant as healthy or diseased, or multiclass predictions that specified the exact type of disease which would allow users to find the best treatment plan for their plants.

Our model architecture was inspired by the hybrid approach presented in Bedi Gole (2021), which combined CAEs and CNNs to balance classification performance with computational efficiency.

The CAE component consisted of multiple convolutional and max pooling layers, using "same" padding to preserve spatial dimensions throughout the encoding process. The CNN portion of the model began with deeper convolutional layers that employed "valid" padding to allow for greater feature abstraction. These are followed by fully connected dense layers that interpreted the abstracted features and made the final prediction, either a binary disease classification or a multiclass disease identification.

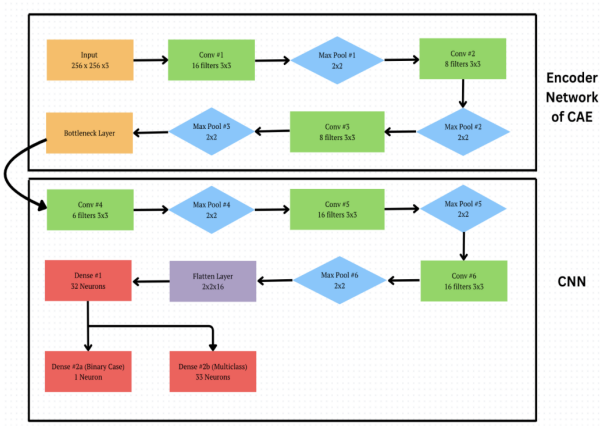

Figure 2: *Diagram representing the architecture of our model. Outputs can be either a binary prediction or a multiclass prediction, based on arguments input by the user before training.*

In total, the hybrid model consisted of 7 convolutional layers, 6 max pooling layers, and 2 dense layers. For the binary classification task, this resulted in just 8,560 trainable parameters, while the multiclass case required 9,583 trainable parameters. These low parameter counts, paired with competitive evaluation metrics, demonstrated the potential of hybrid models for practical, real-world agricultural diagnostics.

3 Results

Model Type	Test Accuracy	f_1 Score	No. of Parameters
Our Binary Model	97.58%	97.58%	8,560
Our Multiclass Model	84%	79 %	9,583
Bedi & Gole (2021) Binary Model	98.38%	98.36%	9,914

Table 1: *Table of results for each model we trained compared with the model from Bedi & Gole (2021). Note that weighted f_1 score was utilized.*

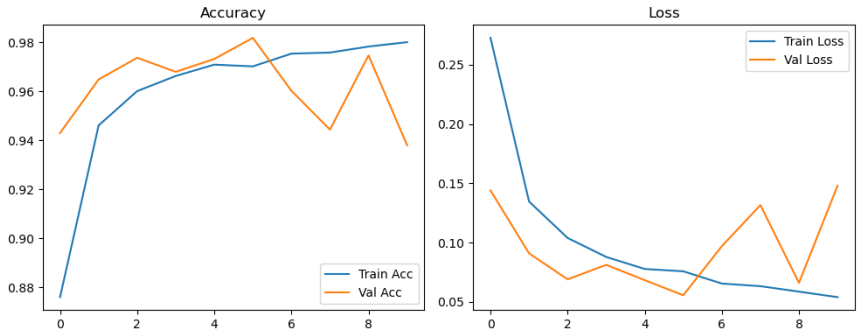

Figure 3: *Accuracy and loss curves for the training and validation sets for the binary model.*

Our binary and multiclass model both had low numbers of trainable parameters compared to the reference hybrid model. As mentioned above, this reflects its computational efficiency and is a key metric for our results as well as determining its success. **Ta-**

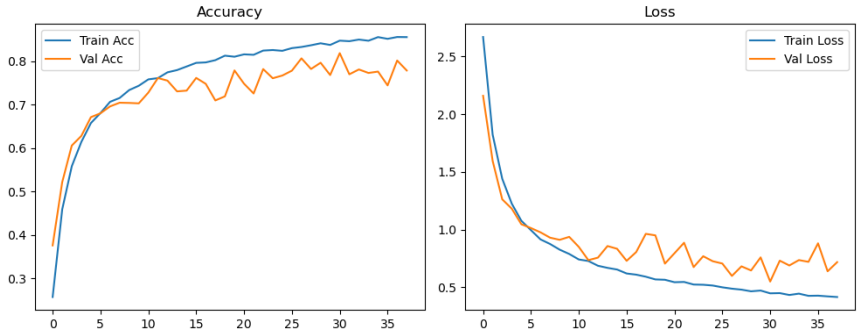

Figure 4: *Accuracy and loss curves for the training and validation sets for the multiclass model. Note that the number of epochs increased compared to when training the binary model.*

ble 1 shows the binary model's 97.58% test accuracy, which was in the higher-end of the range we expected. When compared to the reference paper's, our binary model achieved a 13.6% reduction in the number of parameters with only a 1% reduction in accuracy.

The multiclass model's 84% test accuracy shown below falls within our initial goals of 85%. Similarly, our 79% weighted f_1 score shows the model's ability to perform well on every class. The multiclass metrics hold more weight due to the 33 different possible classes involved, to which binary models pale in comparison. The multiclass metrics could have been improved by further hyperparameter tuning and modifying the architecture; however, due to long run times, even on centralized compute clusters, it would be computationally expensive to optimize hyperparameters using methods such as gridsearch.

4 Challenges

While developing our project, we ran into a few challenges, beginning with curating the dataset itself. The PlantVillage dataset contained numerous incomplete or underrepresented classes, which could have introduced bias or instability during training. To prevent this problem, we made the decision to remove insufficiently populated classes in order to make sure there was a balanced representation of healthy versus diseased examples. For example, the Raspberry class only contained images of healthy raspberry plant leaves with no corresponding diseased examples, so it was excluded to maintain this balance and consistency across the dataset.

Managing the scale of the dataset was another challenge we came across. With 40,000 images, preprocessing required a lot of memory and time, especially when standardizing image resolution and applying data augmentation techniques like random rotations and flips. Making sure that our project was both efficient and scalable proved to be difficult, because we had to balance preprocessing speed with the need to preserve critical image detail and variability for our model's performance.

Lastly, we had to design flexible label structures to support both binary classification for detecting the general health status of a plant and multiclass classification that can be used for identifying specific diseases in plants. For the binary task, we needed to consolidate all diseased classes into a single "diseased" label, creating a clear distinction between healthy and unhealthy plants. In contrast, the multiclass task required preserving all 33 individual disease categories

to enable specific diagnosis. Managing both label formats required careful preprocessing, including precise control over data splits and label mappings to avoid data leakage and maintain class balance. Updating our training and evaluation process to handle both tasks added some extra work, but it allowed us to see how well the model could perform in different real-world situations.

5 Reflection

Overall, we were satisfied with our project and our capacity to meet our goals that we had laid out beforehand, specifically hitting our stretch goals for the binary model and our targets for the multiclass. We were also content with our strong handling of the preprocessing stage, which led to higher accuracy scores with much fewer parameters. The decisions we took to adjust for roadblocks such as incomplete classes or to improve the model such as rotating image inputs were all key to our success. In this sense, we also take pride in the process of the project itself, and our ability to adapt along the way.

During the course of the project, we began to contextualize our model in the real world and think of its potential impacts or use-cases. Given its robust performance on specific plant types and its low number of parameters, we could see how the implementation of this model would excel in farms or large plant-growing operations. Integrated with vision systems to capture photos, for example, a farmer could check on the overall health of strawberry crops in their field. If we had more time, we could try to create a more tailored version of the model to a specific set of crops that farms may grow together, thus simulating this real life use-case. Furthermore, we would have also liked to conduct more hyperparameter tuning and to add more changes to the model architecture. This project taught us a lot about the power of autoencoders for image compression, all while retaining high performance. In fact, the 'portability' and inexpensive nature of the model provoked us to think about how deep learning can be applied at such a small and accessible scale without the need for an internet connection. Indeed, today plant disease recognition is possible with widely available access to cloud computing, but the reference paper explained how these kinds of models use millions of parameters and are thus fundamentally non-local. Going forward, this will inspire us to think about the reach of deep learning techniques beyond the internet and centralized computing.

6 GitHub

https://github.com/rio-wombacher/plants_plants_plants

7 References

Bedi, P., & Gole, P. (2021). Plant disease detection using hybrid model based on convolutional autoencoder and convolutional neural network. Artificial Intelligence in Agriculture, 5, 90–101. https://doi.org/10.1016/j.aiia.2021.05.002