Evan Apinis
12/9/24
ECE 579 Project Report

Table of Contents

1. Introduction/Problem Statement

Apart from my studies and computer engineering work, one of my personal hobbies includes gardening. I have been growing hot peppers and making hot sauces for several years. As a result of this I am fairly familiar with many of the challenges associated with growing peppers, not just weather, but also disease related. Some examples of these are blossom end rot, blight, and pests like aphids or mites. This year however, I tried something new and bought two apple trees from a local nursery, one a honeycrisp, the other a golden delicious.

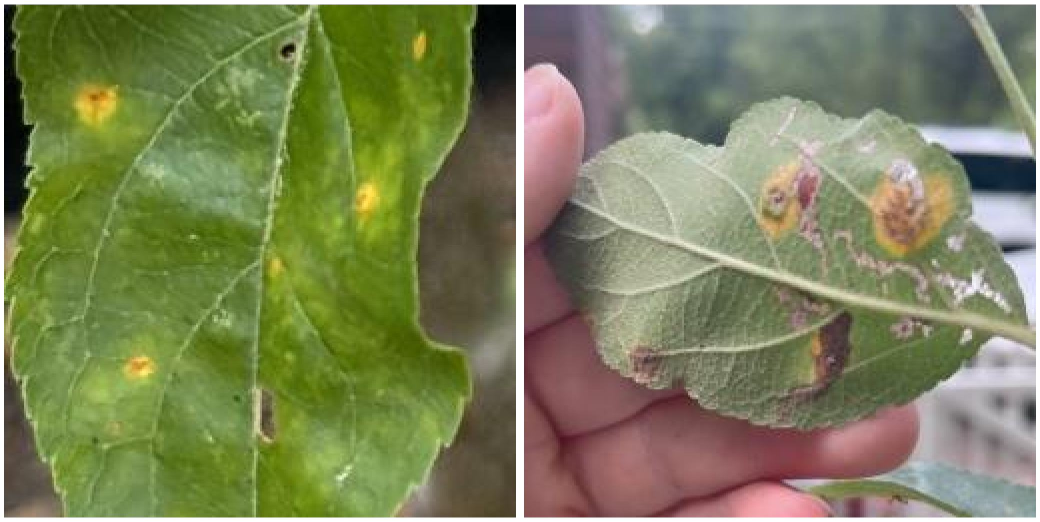
Figure 1 - Picture of diseased leaves on apple tree (Left: May | Right: August)

Not long after planting the trees, they both began to show signs of an infection, with one being worse than the other. The image displayed in Figure 1 shows an image I took of one of the infected leaves during the summer. As I am not an arborist and don't have experience with apple trees I was not able to immediately identify the type of disease without conducting some research first. From my research my best estimation to what it is, and what I treated the trees for this fall, was cedar apple rust. A disease that is spread from nearby infected cedar trees, whose galls opened and infected my apple trees. The purpose of this project is to find a dataset containing cedar apple rust, among other apple tree diseases, and train a model, to which I can give it the image from my tree to verify or dispute my personal identification.

2. Related Work

There has been previous work in the classification of plant diseases from several research teams. Highlighting two of these in particular, one implemented a process similar to my approach using an earlier version of the dataset used for this project. The research conducted in this paper [1] uses the PlantVillage dataset. This dataset contains the same number of classes and species as the dataset used in this project but with two main differences which are explained in the approach section. The PlantVillage dataset is older and does not contain a predefined split of training and testing data. In addition, it is a smaller dataset with less images per class and does not contain the same level of augmentation to the training data, which should lead to lower model accuracy. The paper also uses two different models than the one used in this project, namely AlexNet and GoogLeNet, which achieved a peak accuracy of 99.35%. The second related paper I found [2] used predominantly supervised models, with the exception being K-Means to perform image classification and plant disease identification. The maximum accuracy achieved in this paper by an SVM model was 95.87%.

3. Approach

The first step for this project was to identify an image dataset that would be suitable for my needs. My requirements were as follows.

- Contains cedar apple rust and other apple tree images
- Has a clear organized split of training and testing data
- Contains a sufficient amount of variance in training sets to prevent overfitting

The following dataset, New Plant Diseases Dataset, can be found on Kaggle. The dataset contains cedar apple rust, healthy apple tree leaves, and 36 other leaves from a variety of plant species. Although the dataset covers more than just apple tree diseases, I thought that the addition of other plant species would be interesting. I wanted to see if the model could not only predict the disease, but also the species. The dataset also has a well defined split of training and testing data for all 38 categories. The predefined split for training/testing data was 80/20 with a total of 87k images. The final criteria was to have a sufficient amount of variance in training data. As observed in HW4, without applying preprocessing techniques to the Cifar-10 dataset the ResNet-18 model performance showed overfitting to the training dataset. The new plant diseases dataset is very similar to another dataset on Kaggle, they share the same 38 classes for training and testing. The difference with my dataset of choice is that many of the images in the training set were altered, including image rotation and non-uniform image backgrounds.

The second important decision was to choose a suitable model for training on the dataset. My personal preference was to value absolute performance over computational expense. Through the four homework assignments there was a clear winner for models. Although the unsupervised and supervised models explored had their own trade offs, the residual network ResNet model from HW4 had by far the best validation accuracy on Cifar-10. Therefore for this project I chose ResNet-18 as the model to train with. Although there are alternative complexities for the ResNet models I stayed with ResNet-18 as opposed to

ResNet-50 because the plant disease dataset is still relatively simple and using higher depth models will begin to rapidly fill the memory capacity of my GPU.

The images found in the dataset were all 256 pixels by 256 pixels. When setting up the transform definition for the training and testing data I retained the image sizes, applying a 0.5 mean and standard deviation for normalization. Additionally, rather than set custom weights for the model I used the default ResNet-18 weights found in the torchvision library. The final step to initializing the model was to pass the number of classes in the dataset for adjustment of the final layer, this was done by finding the number of unique directories inside the training folder of the dataset, in total there are 38 classes among healthy and diseased plants.

```
# Define ResNet-18 model
model = resnet18(weights=ResNet18_Weights.IMAGENET1K_V1)
model.fc = nn.Linear(model.fc.in_features, len(diseases))  # Adjust the
final layer for 38 classes

# Move model to DirectML device
model = model.to(dml)
```

```
def train(model, train_loader, criterion, optimizer, device):
```

I then created two helper functions, one to train the model, and one to test the model. The train function is responsible for taking the training dataset loader and feeding the input images and their labels to the model. The model is also defined to run on a dedicated AMD GPU using DirectML, and the training function will apply to dml. Finally the function calculates the running loss of each epoch along with the accuracy using the training dataset. This was included to provide insight to overfitting and underfitting the training dataset.

```
def test(model, test_loader, criterion, device):
```

The test function is close to identical to the train function but handles the model evaluation of the current epoch with the testing dataset. Again, the running loss and model accuracy are calculated to report the performance of the model over each epoch. The train and test functions are then wrapped in a for loop which iterates through the number of epochs specified, for this project I chose ten for reasons shown in the experiments. After completing all ten epochs the model was saved, to allow for loading in a separate python script for evaluation, avoiding the requirement to train the model each time. The separate python script is used to predict labels for the images displayed in Figure 1.

4. Experiments

The first results for this project are the training and testing dataset accuracy results. The table summarizes the accuracy and loss for training and testing across all ten epochs. The table shows that even in the first epoch the training and testing accuracy were very high, 93.70% and 98.84% respectively. This far exceeded the results that I was expecting at the start of this project, but the training and testing loss showed that there was still room for improvement. The following epoch had the largest single epoch gain as the training accuracy jumped to 99.11% with the testing accuracy also surpassing 99% with 99.36%. The loss for training and testing dropped significantly, which mirrors the accuracy gain observed. As mentioned earlier, an epoch count of ten was identified as sufficient given that the training and testing accuracy had begun to level off. It is possible that extending this to 20 epochs would have resulted in marginally better performance, but it had for the most part leveled out with the training loss getting very close to 0. The peak performance of the model was observed in the final epoch where the testing accuracy observed was 99.70%.

Table 1 - Training and testing accuracy/loss per model epoch

Epoch	Training Accuracy	Training Loss	Testing Accuracy	Testing Loss
1	93.70%	0.3159	98.84%	0.0443
2	99.11%	0.0402	99.36%	0.0241
3	99.59%	0.0212	99.32%	0.0219
4	99.75%	0.0142	99.52%	0.0165
5	99.85%	0.0100	99.53%	0.0143
6	99.90%	0.0074	99.61%	0.0126
7	99.95%	0.0054	99.61%	0.0123
8	99.96%	0.0046	99.62%	0.0121
9	99.97%	0.0037	99.66%	0.0119
10	99.97%	0.0033	99.70%	0.0109

The goal of this project for me and my own definitive mark of success or failure was to load the trained model and test it using several pictures I had taken over the year from my own apple tree. I chose two images of the diseased leaves, one from early in the year (May), and one from later in the year (August). Secondly I wanted to test it using a regular healthy leaf. Unfortunately, I did not have any sufficient images to cover this category, although I did still attempt to classify and will share those results, so I found a stock image of a healthy apple tree leaf. Figure 2 shows the images tested on the model, with the label predicted by the model placed on top of the images.

```
for image_path in image_paths:
    # Predict using the loaded model
    predicted_label = predict_image(image_path, loaded_model, dml,
transform, diseases)
    predicted_labels.append(predicted_label)
```

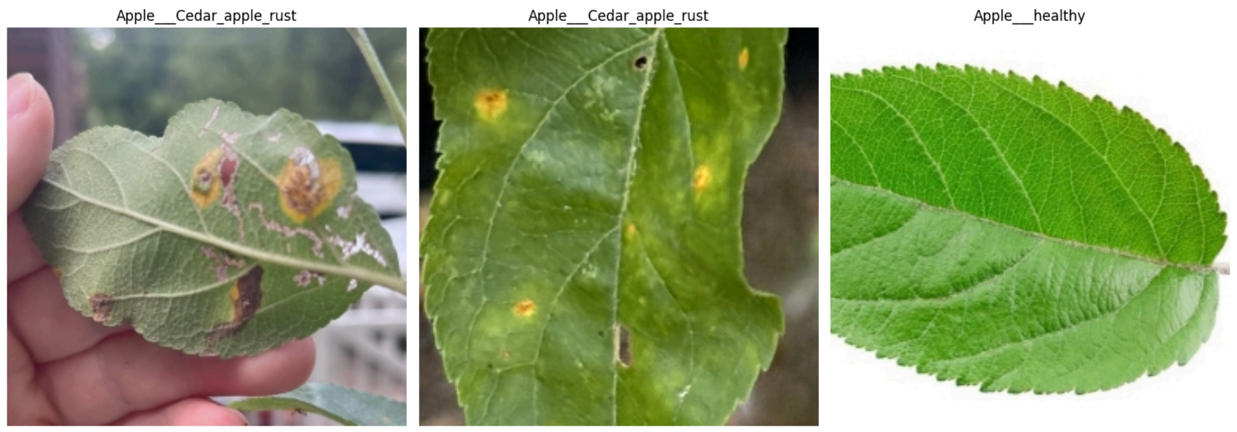

Figure 2 - Test images and the model predicted label

The results show that for the two diseased leaves, which share common features but are in very different stages, were classified as cedar apple rust. This was the original prediction I made after researching the disease, which the model agreed with. Additionally, even among thirteen other types of plants and trees the model was able to correctly label the leaves as belonging to the apple tree species. The stock image was found online for an apple tree and the model correctly labeled the image as a healthy apple leaf. This to me was very successful and backs the 99.70% testing accuracy of the model, but I did still run into issues.

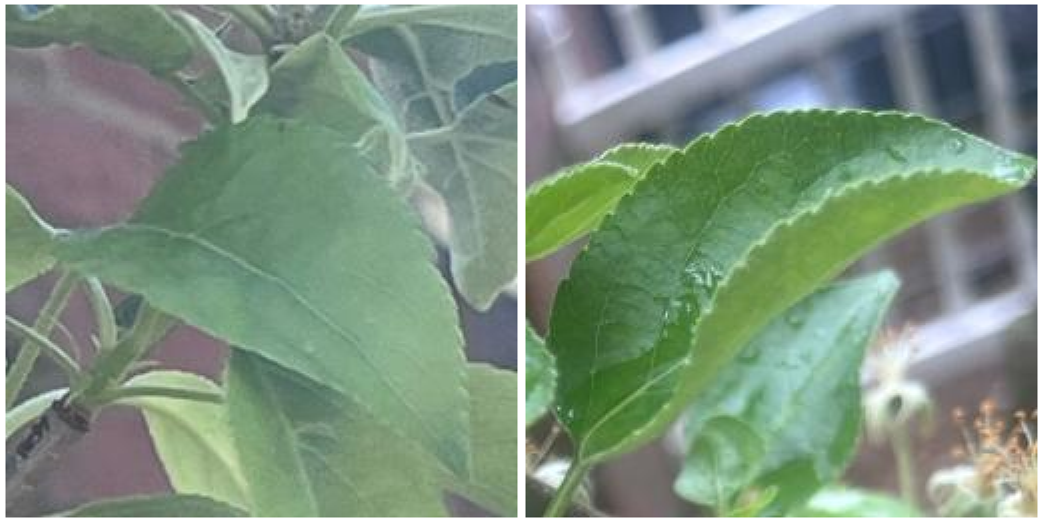

Figure 3 - Test images of misclassified healthy apple leaves

Figure 3 shows two pictures that I had of healthy leaves in the beginning of the growing season for my apple trees. They are zoomed in cropped images from two different photos that I had. When I tried running the model on these images they gave me two different results. For the image on the left, it was classified as a healthy tomato leaf, the right image was classified as a healthy blueberry leaf. I have a few speculations as to why the healthy leaf images I had were unable to be classified correctly. When compared to the images shown in Figure 1 and Figure 2, the images displayed here are not flat. The leaves are curled, on the left the leaf is curled away from the camera revealing only part of the leaf, in addition to a very low quality photo due to the zoom and cropping. For the image on the right the leaf is curled upwards which reveals part of the front and part of the back. Although the training images were augmented, with some rotated and others grey scaled, they were for the most part flat as shown in Figure 4. Additionally, I speculated that classifying healthy leaves may be more difficult than diseased leaves, given that some diseases such as cedar apple rust are unique to apple trees and will not be found on a blueberry or tomato plant.

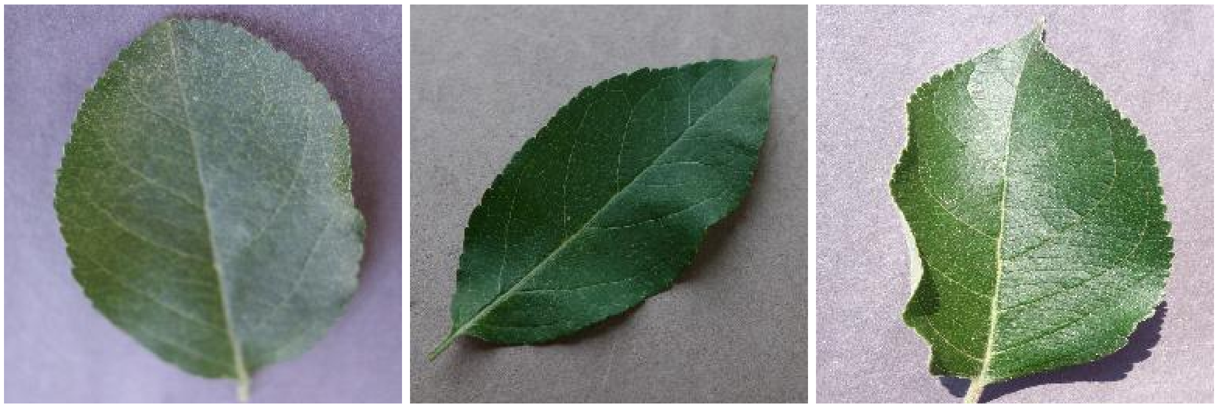

Figure 4 - Healthy apple leaf images from training dataset

To test the hypothesis that healthy leaves were more difficult to differentiate I generated a confusion matrix for the trained model using the testing dataset. The confusion matrix observed for this model is shown in Figure 5. It can be seen that the confusion matrix disputes this claim however. The only labels that the model seemed to misclassify, marginally, was healthy corn and corn gray leaf spot. For these labels we saw a misclassification of 20 gray leaf spot images as healthy corn leaves. Looking across healthy apple tree leaves, healthy tomato leaves, and healthy blueberry leaves, there were no misclassifications, the opposite of what I thought may have occurred. From these results, I attribute the misclassification of my images to poor image quality and/or not enough data diversity in training images.

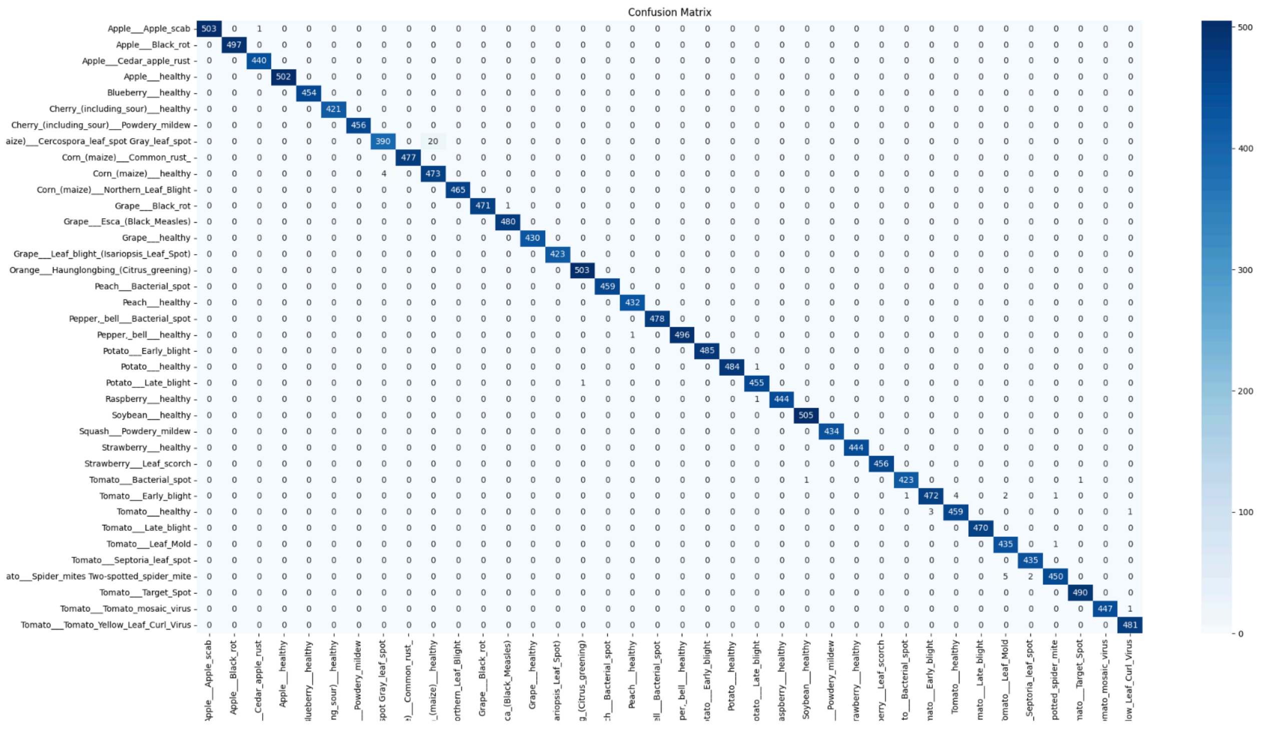

Figure 5 - Confusion matrix for trained model

Table 2 - Classification report for observed misclassified leaves

Leaf	Precision	Recall	F1-Score	Support
Apple Scab	1.00	1.00	1.00	152
Apple Black Rot	1.00	1.00	1.00	150
Cedar Apple Rust	1.00	1.00	1.00	132
Apple Healthy	1.00	1.00	1.00	151
Blueberry Healthy	1.00	1.00	1.00	137
Tomato Healthy	0.99	0.99	0.99	139

5. Conclusion

The goal of this project was to create and train a machine learning model that I could then use to aid in the identification of a leaf disease that infected my apple trees in this past growing season. I was able to find a dataset that contained 38 different classes of leaves, both diseased and healthy, across 14 different species of plants. Notably, the dataset contained cedar apple rust, the disease in question for my apple trees. I then trained a ResNet-18 model on the dataset. To evaluate the model, I supplied it with two different images of diseased leaves I had taken, from different stages, one with subtle features and the other with more pronounced. The model was able to classify both images as the predicted outcome. When it came to classifying healthy apple leaves however, I did run into issues where the model did not perform up to the 99.70% accuracy that was achieved with the testing dataset. To analyze the issues related to this prediction with the model, I generated a confusion matrix and investigated the images in the training dataset. Through the confusion matrix and stock photos I determined that the misclassifications of the model on the healthy images I provided were due to the quality of images supplied, not a common misclassification of the model itself. If in the future I continue to use this model for disease identification, it could be useful to train the models for individual plant species, which may give even greater accuracy results. Although the 99.70% accuracy achieved in testing was able to beat the 99.35% accuracy from the PlantVillage dataset found in this paper [1].

References:

[1]Frontiers | Using Deep Learning for Image-Based Plant Disease Detection

[2]https://www.researchgate.net/publication/337023535_Plant_Disease_Identification_and_Classification_using_Image_Processing