Identifying Powdery Mildew on Plants: Image Recognition and Convolutional Neural Networks

ABSTRACT

Powdery mildew is a disease that can be catastrophic to plants and their surrounding ecosystems. Identifying the disease is the first step toward fighting it. To do this, we trained a variety of convolutional neural networks to perform image recognition and classification. We trained our models using the seminal LeNet Architecture. We tested the impact that increasing the size of the training dataset had on the accuracy of the predictions. We also tested how training models using datasets with different types of images impacts prediction accuracy. We were able to successfully train a neural network that identifies the presence of powdery mildew with 100% accuracy and the absence of powdery mildew with 90% accuracy when tested with new images.

INTRODUCTION

Powdery Mildew and other plant diseases can wreak havoc on plant ecosystems when left unchecked. Powdery Mildew is a plant disease that spreads rapidly by means of asexual and sexual reproduction, and it can do so under harsh environmental conditions. When plants become infected with Powdery Mildew, it often leads to substantial plant loss, lower fruit yields, and associated economic losses for humans. If not treated correctly, its spores will stick around in the local environment and cause problems for a long time to come. Thus, it is essential to identify Powdery Mildew as early as possible in order to treat it and prevent plant loss and other damage [1].

A neural network is a layer of nodes that is designed to function somewhat like a human brain [2]. This is because the types of problems that neural networks are interested in solving are

problems that involve identifying patterns that humans typically do innately, such as image recognition or identifying human handwriting. Nodes in a neural network are interconnected in a one-directional manner, with weights between nodes that determine whether or not data is significant enough to go on to the next level. Neural networks are trained with data, which results in specific weights for node connections and ultimately determine the accuracy of the network [3]. Neural networks are a highly complex form of machine learning, and they can be used for very powerful purposes.

Traditionally, Powdery Mildew has been identified primarily with the human eye, due to its powdery white appearance. While this is effective, it may not be the most efficient solution. Due to its unique appearance and destructive capabilities, powdery mildew is a good plant disease to identify by way of a neural network. Our first step is to train a basic neural network that, when presented with an image, can identify with good accuracy whether or not that image contains a leaf in it. From here, we will train a new neural network specifically aimed at identifying Powdery Mildew on plants.

The experiments we will perform test the efficacy of our trained neural networks. We want to know how well our models work, which in this case means how well our models recognize a leaf or Powdery Mildew on plants, and how modifying certain aspects of our model changes the accuracy of our neural network predictions. For example, we will train models with varying numbers of images and datasets that vary in terms of the images they contain. Then, we will calculate how often our trained models give predictions that match the actual result, and how often they fail.

RELATED WORK

We worked to find articles and related work in two categories, the first being related deep learning applications, and the second being resources that would help us to implement and train convolutional neural networks. Overall, this literature helped us to find the best way to attack our specific problem. First off, we found an article measuring fruit maturity, as well as another trying to figure out the readiness of maize.

Fruit maturity is extremely important for the economy of the Philippines. Ayllon, et al. trained multiple neural networks to take images of one of three different fruits and tell whether the fruit is premature, mature, or over-matured. The researchers selected 60 pieces of fruit per day and took 20 images of each fruit from different angles. From this data, they trained the neural network in order to be able to make predictions for new fruit. Their neural networks worked the best for predicting the maturity of bananas (96% accuracy), and worst for predicting mangos (85% accuracy). The neural networks trained with images of color and performed significantly better than the networks trained on images that were grayscale. To improve future models, the researchers suggest training models with more images, with more angles, and with varying distances from the camera [4].

Maize is a staple in the diet of Nigerians, no matter the dish. However, there is also a burgeoning desire to produce maize to feed livestock, but the only problem is that maize leaves contain no nutritional value at peak maturity, which is when humans typically consume maize. Rather, maize leaves are most nutritional before the plant is biologically mature. These researchers did not trust the consistency of human eyes to differentiate between precise colors of maize, so they created a neural network to identify when a maize plant is optimally mature to feed livestock. They take their own pictures, which they divide into a training set, a validation set, and a testing

set. They find that their trained neural network accurately predicts when maize is optimally matured to feed livestock 100% of the time [5].

Alternatively, there were a couple papers that helped us form our code significantly [6]. This article gives an in-depth, technical description of how to train a simple neural network using Python and Keras. The neural network that is trained in this demonstration identifies images that are of santa compared to images that are not of santa. It is pretty arbitrary, but it serves as a good foundation, and became what our leaf vs. not leaf neural network was almost entirely based on.

Additionally, facial recognition seemed to be quite similar to our goal [7]. This paper addresses the problem of gender recognition from face images taking into account the memory and the running time issues by using a small training dataset which makes it possible even on small devices with a limited memory and without dedicated graphical processors for computations. It uses a publicly available dataset which makes it easy to follow along and relatable to our project. The research manages to Minimize the input image size and associated number of convolutional layers, number of filters and size of the fully connected layer. The CNN recorded a performance of 97.31% using the ensemble of 3 layers.

Finally, to improve our knowledge of CNN models we found an article looking to increase the accuracy of a CNN model by introducing a hybrid model of CNN and a General Regression Neural Network [8]. With powerful function approximation to recognize images according to extracted representations the hybrid model strengthens the effectiveness. However the paper further suggests that a CNN model that contains a large number of weights ultimately means a large number of iterations, so it is going to be time consuming. As the accuracy increases the

time taken to get the desired result also increases. This is good information to know when it comes to our project.

APPROACH

When creating a convolutional neural network that classifies images into two distinct categories we quickly learned that without enough training data, a deep learning and machine learning model can't learn the underlying, discriminative patterns required to make a robust classification. Taking this into consideration after doing some research we found out that "there is a way to leverage the power of google images to quickly gather training data and therefore cut down the time it takes to build a dataset" [6]. we adapted a function that manually intervenes in javascript and simulates clicking on an image in the browser by dispatching a mousedown and a mouseup event. We defined a function that is able to extract the url and store it in a query string which meant that we could save all the URLs loaded on the browser for each image onto a text file. After that step downloading the images wasn't very hard to do. This was the main process we took to get training images for our deep learning model.

Once we explored the acquisition of our training images we moved on to training the seminal LeNet Architecture. A seminal LeNet architecture is a desirable "first image classifier" for convolutional neural networks, "... originally designed for classifying handwritten digits, we can easily extend it to other types of images as well" [6]. An LeNet architecture is a fairly small network, that contains the basic modules of deep learning. It has two sets of convolutional, activation, and pooling layers, followed by a fully-connected layer and finally a softmax classifier. In this project we will be implementing this architecture using Keras and Python.

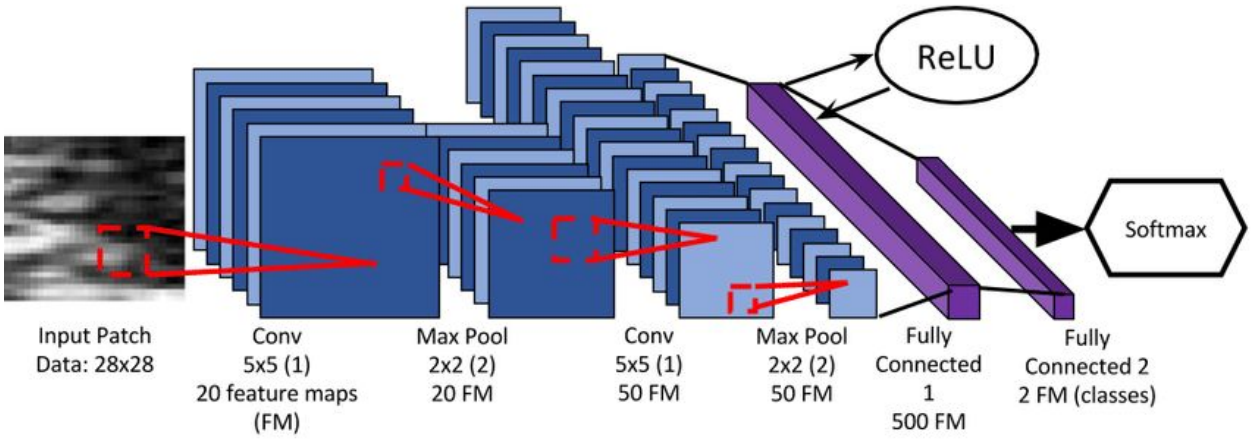


Figure 1: LeNet Architecture

After acquiring a majority of our source code from [6] we made small changes to the functions that are responsible for building and testing an effective deep learning model. We adapted a file that defines our network architecture with a class called LeNet. An image is a multidimensional matrix, which means that the width of an image can act as columns to a matrix and the height of an image can act as rows. Once our model was initialized we started adding CONV Layers. Convolutions are an element-wise multiplication of two matrices followed by a sum. First it takes 2 matrices (In our case a 5x5), then it's able to multiply them, element by element(not by dot product and finally it sums the elements together to get the desired result.

CONV layers compute the output of neurons (matrices) that are connected to local regions within the input, in other words these layers are small pieces of the bigger matrix. Afterwards the RELU and POOL Layers are added. The final block before training our model is flattening out the volume into a set of fully connected layers as shown in figure 1. In the process of deep learning and convolutional neural networks it's important for datasets to have a training (sample of data used to fit the model), validation (data that provides an unbiased evaluation of a model fit) and testing (data used to provide an unbiased evaluation of the final model) split ratio. This ratio is

fully dependent on the number of samples and the actual model being trained. After setting up the architecture of our deep learning model we performed a training and testing split, using 75% of the total dataset for training and 25% for testing. We chose to allocate only the two splits because after some research we found out that 75/25 is usually the typical testing split for a small amount of data. Finally with the use of matplotlib we were able to plot our results and successfully analyze how our deep learning image classifier is performing for each trained model (See figure 4 and 9). This was mainly our process to build the Leaf not Leaf model. After building the Leaf Not Leaf model we were able to collect different images of Powdery mildew on leaf and train that against other healthy leaf images to finish the technical part of our research.

EXPERIMENTS

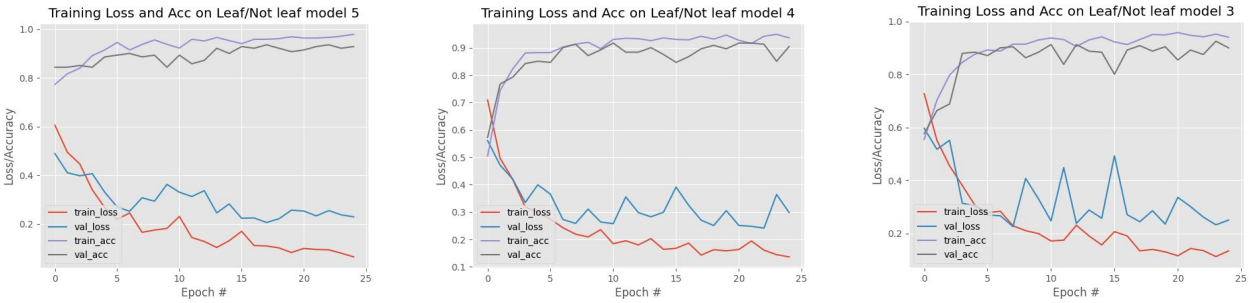
When creating our individual models, we decided to keep a few things constant throughout. While performing a training and testing split as part of our deep learning model we used 75% of the total dataset for training and 25% for testing. This is usually the typical testing split for this amount of data. To test our models, we used two different types of datasets. The first dataset [D1] we found was a specific dataset from Mendeley Data [9], that had images of leaves in a very controlled setting - black background, perfect focus, all healthy leaves. Our second dataset [D2] was a more general one we gathered from Google Images, showing far more variety in color, setting, size, and type of leaf.

Figures 2a and b: examples of D1

Figures 3a, b, and c: examples of D2

Leaf Identification

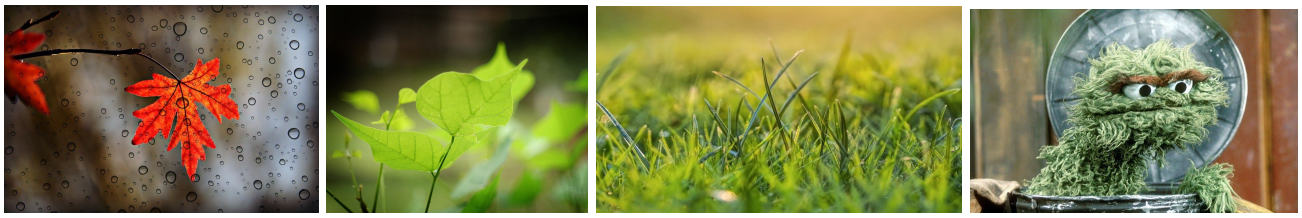
Our first experiment focused on solely identifying if there was a leaf in an image. We wanted to focus on the size of the training dataset, to identify those effects, so we used a variety of images from both D1 and D2, and used three different sizes of datasets: 100 images, 500 images, and 961 images. We'll call these three models Model100, Model500, and Model961.



Figures 4a, b and c: trained model plots for Model100, Model500, and Model961 respectively

Once each dataset had trained the model, we used the same set of 10 leaf images and 10 not_leaf images to test. Within our leaf images were a variety of leaves: different colors, different

backgrounds, even a drawing of a leaf, all from outside of the training dataset, just to identify how strong the leaf identification was.



Figures 5a, b, c, d: examples of leaf and not leaf testing images

Alternatively, within our not_leaf images were images attempting to trick the model, as seen in figures 5c and 5d. We had a tree with no leaves, a single branch, and then a variety of other green images: green apples on a wooden background, Oscar the Grouch (leaflike fur), Kermit the Frog (leaflike neckpiece), and so on. These not_leaf images could be anything, so we just wanted to choose ones that would hopefully give us some interesting results, and it certainly did.

		Leaf										Not_Leaf									
	Image	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Model100	**Leaf?**	0	1	0	1	0	1	1	1	1	1	1	0	1	0	0	0	0	0	0	1
	Percent Confident	99.5	53.1	88.4	96.6	99.8	98.9	98.1	100	99.9	96.2	98.9	99.9	99.1	64.4	88.2	61.9	71.3	92.9	99.9	60.9

Figure 6: Model100 results

After testing Model100, we found that 7 of the 10 leaf images were identified correctly, and an additional 7 of the 10 not_leaf images were identified correctly. Looking at figure 6, each image was around 90-100% confident, whether correct or incorrect, with a few exceptions. The things that seemed to throw the model off when identifying leaves were red leaves (leaf images 1, 3, and 5), more zoomed in images (leaf image 5), and our one leaf drawing (leaf image 2). For the drawing, it still identified it as a leaf, but was not very confident. Alternatively for not_leaf, it was thrown off by images featuring grass (not_leaf images 1 and 10).

		Leaf										Not_Leaf									
		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Model500	**Leaf?**	0	1	1	1	0	1	1	1	1	1	1	0	1	1	0	1	1	0	0	1
	Percent Confident	99.7	99.6	66.5	66.4	50.9	99.9	100	100	100	99.9	100	99.9	99.9	96	52.2	81.4	94.9	88.4	94.5	99.3

Figure 7: Model500 results

Testing Model500 showed us more accuracy when identifying leaves, but became far less accurate when identifying things that weren't leaves. In this model it correctly identified 8 of the 10 leaves, but only 4 of the 10 random images, as seen in figure 7. Leaf image 3 was now correctly identified as a leaf, but leaf images 1 and 5 were still identified as not leaves, though image 5 was far less confident, which was an improvement. Additionally, the leaf drawing identification was far more confident in this model. Looking at the not_leaf images, they appeared to be thrown off by images with green colors and grassy shapes (not_leaf images 1, 4, 6, 7, and 10).

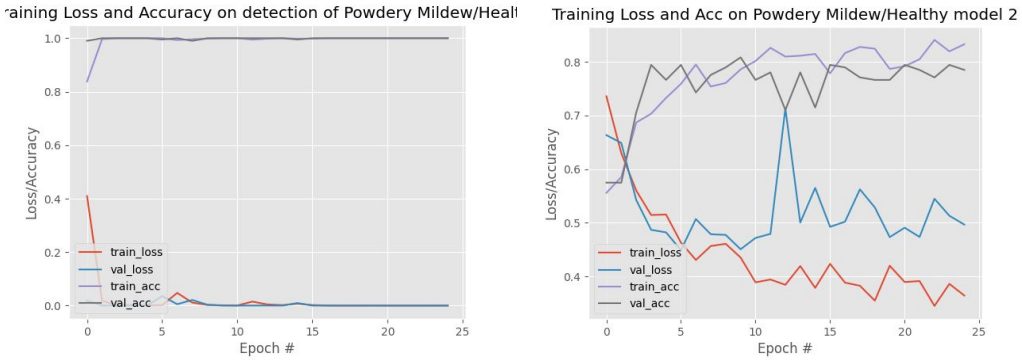
		Leaf										Not_Leaf									
		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Model961	**Leaf?**	0	1	1	1	1	1	1	1	1	1	1	0	1	0	1	0	1	0	0	1
	Percent Confident	91.2	96	69.2	82.3	83	95.8	99.8	100	100	99.8	99.4	83.5	98.5	50.9	72.9	64.7	58.8	72.8	87.1	94.3

Figure 8: Model961 results

The tests for Model961 improved on Model500 for both leaf and not_leaf identification, but not_leaf identification in Model100 was still more accurate. 9 of the 10 leaf images were identified as leaves, while 5 of the 10 not_leaf images were identified correctly. Leaf image 5 was now identified properly, and otherwise overall the percent confident was increasing, which is a good sign. There was more flipping in terms of accuracy for not_leaf images. Not_Leaf images 4 and 6, Kermit and a dragon, were back to being correct, while image 5, of a branch, was now incorrectly identified. The percentages all seemed to lower, so no matter if their guess was accurate or not, the model was overall less confident in its results.

Powdery Mildew Disease Identification

The final experiment that we performed tested the efficacy of the two different neural network models that we trained to detect whether or not an image of a plant has powdery mildew on it. The first neural network, Model1, was trained off of a collection of images of plants with powdery mildew (found on Google images) as well as images in D1 for non-powdery mildew plants. The second neural network, Model2, was trained off of the same powdery mildew images as Model1 as well as images in D2 for non-powdery mildew plants.



Figures 9a and b: trained model plots for Model1 and Model2 respectively

Specifically, we were interested in testing how well these two models correctly identify the presence of powdery mildew and also how well they correctly identify the absence of powdery mildew. Across the two models, we tested for how the specificity of the training data affects the accuracy of the predictions — since the non-powdery mildew images for Model1 are more similar to each other, while the non-powdery mildew images for Model2 incorporate a wider range of image types (in terms of plant type, color, background, etc.). To test each of the models, we had them make predictions for ten new test images of plants with powdery mildew and ten new test images of plants without powdery mildew.

		Powdery_Mildew										Not_Powdery_Mildew									
		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Model1	**Leaf?**	1	1	1	1	1	1	1	1	1	1	0	0	1	0	1	0	1	1	0	1
	Percent Confident	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	88.7	100	100	100

Figure 10: Model1 results

In testing Model1, we found that it correctly identified powdery mildew on plants 100% of the time, with 100% confidence for all predictions. On the flip side, Model1 correctly identified the absence of powdery mildew on plants only 50% of the time, with an average confidence of 100% for correct predictions and an average confidence of 97.4% for incorrect predictions.

		Powdery_Mildew										Not_Powdery_Mildew									
		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Model2	**Leaf?**	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	1
	Percent Confident	84.8	95.9	98.8	93.5	93.1	76.5	50.3	98.3	98	82.8	66.2	67.2	79.6	66.2	99.7	99.9	76.1	98.4	98.5	63.4

Figure 11: Model2 results

In our test for Model2, we found that it correctly identified powdery mildew on plants 100% of the time, with an average confidence of 87.2% for each prediction. Additionally, Model2 correctly identified the absence of powdery mildew on plants 90% of the time, with an average confidence of 81.5% for each prediction. The one incorrect prediction made by Model2 for non-powdery mildew was 63.4% confident, which was the lowest confidence value yielded for non-powdery mildew test images for Model2.

ANALYSIS

The results from our first experiment on leaf identification demonstrate some nuanced details about our neural networks and the effect that increasing the size of the training dataset has on model performance. Our results show that as the size of the training set increases, the predictions for the leaf predictions do improve substantially. This is significant because it demonstrates that

adding more training images gives the neural network more exposure to more possible inputs, and thus allows it to make better predictions on new test images that the model has not encountered before. One interesting thing to follow is the improvement of leaf images 3 and 5 (figure 12a, 12b). Initially, in Model100, these were both classified as not leaves, with 88.4% and 99.8% confidence levels respectively. We believe this was because of the color and how zoomed in image 5 was - Model100 hadn't seen enough of these differences. However, as the model size increased in Model500, their identifications improved dramatically. Image 3 was now identified properly as a leaf with 66.5% confidence, and image 5 was still not classified as a leaf, but the model's confidence in this was significantly lower, at 50.9%. Then, once we used Model961, both images were finally identified accurately as leaves, with higher confidence levels of 69.2% and 83%. This shows us how the increased training dataset size really benefits leaf identification.

Figure 12a and b: leaf images 3 and 5 respectively

The not_leaf predictions did not improve in a similar manner due to the fact that our testing images were chosen in an attempt to trick the model with image attributes similar to leaves such as color. There are so many possible images to account for in the not_leaf image dataset that it is difficult to account for every possibility. For example, even by increasing the size of our not_leaf dataset from 100 to 500 to 961 images, none of the models had ever encountered a picture of green apples before. But they had encountered many images of green leaves that had been

classified as leaves. Conversely, they had encountered many images of things like skyscrapers and cityscapes that were classified as not_leaf, and to which green apples bore little resemblance. Thus, to identify any possible image as not_leaf becomes more difficult for fringe cases since there are so many possibilities for what can be classified as not_leaf.

The second experiment comparing Model1 to Model2 demonstrates the significance, in terms of prediction accuracy, of training neural networks with diverse training datasets. The two models were trained on the same powdery mildew images; however, the training data for non-powdery mildew plants was different across the two models, and it showed in the prediction results. Model1 was much more suited for a specific type of image of a leaf when predicting for non-powdery mildew, namely, a green leaf at a consistent distance away from the camera and a dark background. Thus, Model1 performed well when tested with images of this nature. However, Model1 became too accustomed to looking for these specific features in the image that it was actually less focused on identifying the powdery mildew. Conversely, Model2 was much more accurate in identifying when leaves did not possess any powdery mildew on them due to the fact that it was trained with a more diverse set of non-powdery mildew images. These plants varied in type, color, background color, distance to plant and more. Thus, Model2 was better equipped to look specifically for the presence or absence of powdery mildew. We see the confidence levels in our predictions drop from Model1 to Model2 because Model2 acknowledges and accounts for more uncertainty in image types.

While we were able to build comprehensive neural networks with fairly accurate predictive powers, we are conscious of some potential limitations of our models. First, we think that our predictions could have been even more accurate had we better controlled for differences unrelated to powdery mildew between images in our powdery mildew dataset vs. our

non-powdery mildew dataset. Differences of this nature include camera angle, distance to plant, background color, etc. We did not have the time or resources to gather the data necessary to train these models with our own photos or control for more variables accordingly. Next, again due to the resources at our disposal, including time, it was not possible for us to have larger test sets when running the experiments that tested our models. If we could increase the sample size of the testing datasets, we would likely get testing results that are even more accurate.

CONCLUSION AND FUTURE WORK

While powdery mildew is a plant disease that can spread rapidly if left unnoticed or untreated, our experiments demonstrate that training neural networks to identify powdery mildew on plants is possible to do with a high level of accuracy. We were able to accomplish this using convolutional neural networks trained specifically with this goal in mind. While we experimented with intermediary models along the way to test how the size of the training dataset impacted the accuracy of the predictions, our ultimate goal was to predict the presence or absence of powdery mildew on plants. We found that this could best be accomplished with a diverse, robust set of training images both for plants with powdery mildew and plants without powdery mildew.

We suggest that future research be done in conjunction with the limitations acknowledged in our Analysis section. Namely, we suggest that experiments be performed in training a powdery mildew neural network with all images taken in the same fashion to control for other variables, and then comparing said model to our own. Further, we suggest that more images be used to test the efficacy of the model. We believe that these additions would add value to the work we have already done. Lastly, future work that we would like to carry out involves training one single

neural network model to not only recognize powdery mildew on a plant, but also be able to recognize other plant diseases. Ideally, then, this one model could distinguish between many different plant diseases and tell us which disease, if any, the plant is likely to have.

REFERENCES

[1] Glawe, D.A; G.G. Grove, "Powdery Mildew Diseases." Pacific Northwest Plant Disease Management Handbook. 2020. https://pnwhandbooks.org/node/387/print.

[2] Laine, Andrew. "Neural Networks." ACM Digital Library | Encyclopedia of Computer Science, January 2003, dl.acm.org/doi/10.5555/1074100.1074638.

[3] Hardesty, Larry. "Explained: Neural networks." MIT News, 2017. https://news.mit.edu/2017/explained-neural-networks-deep-learning-0414.

[4] Ayllon, Mark Anthony; Melwin James Cruz; Justin Jason Mendoza; Mary Christine Tomas, "Detection of Overall Fruit Maturity of Local Fruits Using Convolutional Neural Networks Through Image Processing." ACM Digital Library | ICCBD 2019: Proceedings of the 2nd International Conference on Computing and Big Data, October 2019, dl.acm.org/doi/10.1145/3366650.3366681.

[5] Peter, Ayuba; Sa'adatu Abdulkadir. "Application of Image Processing and Neural Networks in Determining the Readiness of Maize." ACM Digital Library | ICMLSC '18: Proceedings of the 2nd International Conference on Machine Learning and Soft Computing, February 2018, dl.acm.org/doi/10.1145/3184066.3184068.

[6] Rosebrock, Adrian. "Image Classification with Keras and Deep Learning." PyImageSearch, 11 December 2017,

www.pyimagesearch.com/2017/12/11/image-classification-with-keras-and-deep-learning/. Accessed 17 October 2020.

[7] Antipov, Grigory; Sid-Ahmed Berrani; Jean-Luc Dugelay. “Minimalistic CNN-based ensemble model for gender prediction from face images.” Pattern Recognition Letters, Volume 70, 15 January 2016. https://www.sciencedirect.com/science/article/pii/S0167865515003979.

[8] Zhang, Jiajia; Kun Shao; Xing Lao “Small sample image recognition using improved Convolutional Neural Network” August 2018 sciencedirect.com/science/article/pii/S1047320318301810.

[9] Chouhan, Siddharth Singh; Kaul, Ajay; Singh, Uday Pratap; Madhav Institute of Technology and Science (2019), “A Database of Leaf Images: Practice towards Plant Conservation with Plant Pathology”, Mendeley Data, V1, doi: 10.17632/hb74ynkjcn.1

[10] Nelson, Dan. “Image Recognition in Python with TensorFlow and Keras.” Stack Abuse, stackabuse.com/image-recognition-in-python-with-tensorflow-and-keras/ Accessed 17 October 2020.

[11] Singh, Himanshu. Practical Machine Learning and Image Processing. Apress, 2019, link.springer.com/book/10.1007/978-1-4842-4149-3.

[12] Sollami, Mike. “Google Image Search Hack.” Michael Sollami, 14 July 2019, www.sollami.ai/code/2017/1/3/google-image-scraper. Accessed 15 October 2020