Automated Crop Disease Classification: Examining Lightweight Transfer Learning for the Edge

Bhagirath Mehta
Stanford University
Computer Science Department
bmehta18@stanford.edu

Pranav Sriram
Stanford University
Computer Science Department
prsriram@stanford.edu

Abstract

Due to the damaging effects of anthropogenic climate change, automated crop disease identification is extremely important for farmers around the world to utilize existing resources wisely. While recent research has involved the use of SVMs and some CNN approaches for this task, there remains room for improvement, especially when it comes to generalizing classification of healthy and diseased leaves across all species. We tested the effectiveness of using a lightweight custom deep CNN for this task. Additionally, while transfer learning has lent promising results on numerous classification tasks, it has not yet been applied to automated crop disease identification. This paper explores the effectiveness of pretraining on a low-quality plant leaf dataset and finetuning on a high-quality leaf dataset for fine-grained binary disease classification. Our custom architecture was extremely effective with an accuracy of 86.21% *on the low-resolution test set (beating out a baseline model with three times as many parameters). On the other hand, we found that the transfer learning schema did not perform effectively at inference, yielding a test set accuracy of 52.77% on the high-resolution test set.*

1. Introduction

As the world's population continues to grow, increased food production is needed. The United Nations estimates that an additional 60% increase in crop output will be required to feed the world's population by 2050. [5]

With that said, pests and diseases lead to a 20-40% loss in food production around the word [2]. In recent years, due to climate change, this problem has gotten even worse with a decrease in usable crop land alongside a rise in crop pandemics.

As a result, now more than ever, it is vital for farmers to be able to tell whether their crops are healthy or diseased in order to make decisions around what to grow, where to grow, and what other resources, like pesticides, are needed. Currently, this can only be done with constant monitoring of plants throughout each plot of land. In particular, diseased plants typically show signs of disease in their leaves. However, given the scale of this problem, it is not feasible for most farmers to manually check plant leaves around a farm consistently.

Thus, there exists a clear need for a light-weight automated solution to evaluate plant health. Such a system would be run on small cameras placed around crops and would regularly take pictures of leaves and analyze them to predict if the crops show signs of disease. Given the ubiquity of cheap IoT devices, it is economically feasible to leave a network of devices around a farm constantly capturing data, especially if the model used is efficient enough to run on the edge [9]. Thus, it is vital to build a lightweight model that can accurately differentiate between diseased and healthy plant leaves [2].

Our problem is how to best classify healthy/diseased leaves using a lightweight computer vision approach. Our inputs will be JPEG images of a single leaf. We then use a CNN to output a predicted classification of whether the leaf is healthy or diseased. This is a binary classification task.

The variability of plants, leaves, backgrounds, and different stages of growth makes this problem quite difficult to solve, especially given that our goal is to develop a single model and approach that generalizes to all plant species. Given this variability, in this paper, we will explore the effectiveness of transfer learning for plant leaf disease classification. Transfer learning is a subfield of machine learning that uses knowledge from one domain to improve performance in a second, possibly different domain. More specifically, we will investigate whether pre-training our models on a low-quality (but high-quantity) plant leaf image dataset and then fine-tuning on a high-quality (but low-quantity) second leaf dataset containing few overlaps in species or diseases leads to promising results. We chose this approach as we believe it is realistic, given that higher quality data can be harder to come by, but there is more likely to be a

ubiquity of lower quality images. In order to consider the adequacy of this method, we compare the performance of this transfer learning approach to several other experiments involving training and testing on the same dataset, using different architectures and how a model performs without finetuning on the target dataset.

2. Related Work

In regards to the task of classifying plant leaves as healthy or diseased, we have examined several readings for context and background. [6] discussed how using sophisticated neural networks, and many image preprocessing techniques like noise removal, region cropping and image segmentation, along with a very large dataset allowed for a very high accuracy after training on GoogLeNet. However, this takes much manual curation of the dataset, while we would much rather ensure our results are generalizable without requiring much manual curation.

2.1. Crop Classification

Our readings also provided context for useful methods in setting up our classification pipeline. On the classification side, papers such as [9], [2] and [6] provide non-neural network based approaches for classifying crops from satellite images such as decision trees, naive Bayes, support vector machines (SVMs) and logistic regression. While this is slightly different from what we are working on, the papers nevertheless provided useful background information on the current state of the field and the accuracy provided by such approaches, which approached 90%, a high amount given that these papers chose to not classify using just the binary label of healthy and diseased, but actually assigned a separate label for every type of leaf and disease. Of course, this may also prove an easier task in some ways, given that it is not necessarily true that all healthy leaves and all diseased leaves share similar features, and thus, separating out leaves by different types of species and diseases may be a simple task that requires less work than generalizing the task of classification of healthy and diseased leaves across all plant species and diseases.

2.2. Transfer Learning

In regards to transfer learning, we investigated several approaches. In particular, we took inspiration from the work of Elmahdy et al. who found that pretraining on AlexNet, then removing the last layer, allowed for improvement of the overall task of classifying three skin lesions [4]. The key finding of this paper was that despite the fine-grained nature of the task, the pre-trained embeddings generated by AlexNet, along with some finetuning on the specific domain allowed for far superior results to existing methods.

In addition, Qiu et al. reference the same problem that we attempt to address in our paper – the problem of high amounts of low-quality data [10]. Qiu et al. also are reluctant to use many hand-selected features, an approach that does not scale well. Qiu et al. instead use a few different approaches, including artificially increasing the resolution of images to use for 'pre-pre-training' and then augmented the original images for 'pre-training'. In addition, they use 'squeeze-and-excitation' networks to allow different features to be weighted differently. They found that these approaches allowed them to outperform more complex CNN architectures that ultimately required much more data and data that was higher-quality. This approach inspired us to create our own simpler architecture, which would take less data to train, and to use image augmentation on the data we did have in order to improve our pre-training.

We also recognized the possibility that transfer learning does not necessarily transfer very useful embeddings, but ultimately it could speed up convergence, requiring less time to finetune on different species of leaves, as discussed by Aji et. al's work on analyzing what transfer learning actually transfers. [1].

A survey paper on transfer learning by Zhuang et al. discussed 40 different approaches to transfer learning, including homogeneous transfer learning, where both sets of data are in the same domain, and inductive transfer learning, where both sets of data are labeled. [12]. The paper emphasized the benefit of feature-based asymmetric approaches for transfer learning, where features were extracted from a source dataset and transformed into the desired features for the target dataset, which is how we proceeded to conduct our experiments with transfer learning, by first learning our features from the initial dataset, then finetuning them on a second dataset.

Finally, Li et al. discuss the perils of transfer learning in domains with even minor differences, as a massive amount of data may be needed for the finetuning stage [7]. This turned out to be prescient given the ultimate direction of our transfer learning experiment.

3. Methods

3.1. Model Architecture

For this project, we built on top of the work done in [2] by experimenting with other architectures beyond SVMs. In particular, we experimented with CNN approaches (using AlexNet as inspiration) for leaf classification.

Our baseline model consisted of two CONV-POOL blocks followed by two Fully Connected layers as seen in Figure 1.

Our lightweight custom deep CNN (inspired by AlexNet) consisted of the following architecture as seen in Figure 2.

It's extremely important to note that our lightweight custom architecture actually contains far fewer parameters

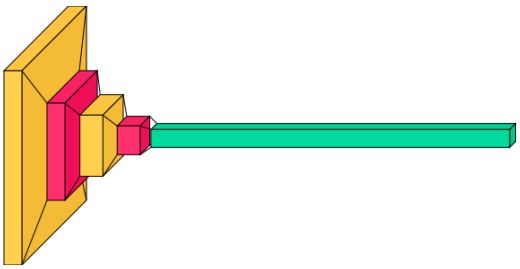

Figure 1. Baseline Model Visualization: Our baseline model consisted of two CONV-POOL blocks, followed by two Fully Connected layers with Dropout. ReLU was used as an activation function. This model consisted of 74,220,858 parameters. The fully connected layers have been omitted for clarity.

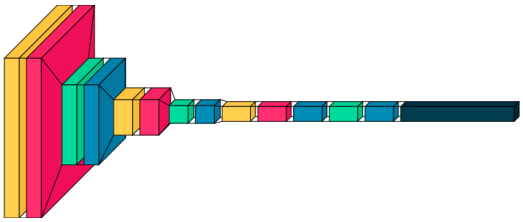

Figure 2. Custom Model Visualization: Our lightweight custom deep CNN model consisted of three CONV-POOL blocks, followed by three Fully Connected layers with Dropout. Leaky ReLU was used as an activation function for the CONV-POOL blocks and ReLU was used as an activation function for the Fully Connected layers. Batch Normalization was also applied across layers. This model consisted of 24,031,362 parameters. The fully connected layers have been omitted for clarity.

(at 24,031,362) compared to our baseline architecture (at 74,220,858) despite being far deeper. In short, our experiments investigated whether a deeper model with fewer parameters could outperform a shallower model with far more parameters. This was intentional, as stressed in our introduction, because it is critical for crop disease detection algorithms to be lightweight enough to run efficiently on the edge. This is also why we decided not to experiment with more complex deeper models (such as ResNet-152) which would not meet the latency requirements and memory constraints needed for on-edge inference.

3.2. Learning Algorithm

For our learning algorithm, we utilized mini-batch gradient descent with momentum. Mini-batch gradient descent works by splitting the training data into smaller batches (in our case of size 128). Instead of backpropagating across a single example (stochastic gradient descent) or across the entire training dataset (batch gradient descent), we backpropagate across each of these mini-batches.

Momentum is a physics-inspired update technique where our gradient affects our model's rate of velocity. This often leads to fast convergence on deeper networks. Here are the relevant formulas for momentum [11]:

$$v = \mu * v - lr * dx$$

$$x+ = v$$

In these formulas, note that v refers to our model's velocity, μ refers to our momentum, lr refers to our learning rate, and dx refers to our gradient value.

For our loss function, we utilized Binary Cross Entropy loss, defined in Figure 3.

$$L_{BCE} = -\frac{1}{n}\sum_{i=1}^{n}(Y_i \cdot \log \hat{Y}_i + (1 - Y_i) \cdot \log (1 - \hat{Y}_i))$$

Figure 3. Binary Cross Entropy Loss: BCE loss operates by penalizing predicted class probabilities that do not correspond to the actual/true label.

3.3. Methodology

3.3.1 Experiments for Determining Architecture

In terms of our methodology, we first ran two experiments – one of which used the baseline model and the other of which used the custom model, in order to determine whether a deeper model with fewer parameters can beat the performance of a shallower model with three times as many parameters. Based on the results from this stage of the process, we decided which architecture was optimal to continue using for our other experiments. The metric we used to determine which model learns the best was the testing accuracy on the PlantVillage dataset.

Next, given the fact that we have chosen an architecture, we can compare three ways of training for the task of classifying images in PlantLeaves, a dataset that has few classes in common with PlantVillage.

3.3.2 Experiments for Transfer Learning

We can first train on PlantLeaves directly and then test on it. Alternatively, we can train solely on PlantVillage and then test our results on PlantLeaves. This would directly test how well the embeddings from PlantVillage carry over to PlantLeaves, and help us to examine how well transfer learning without fine tuning performs, since ideally we could generate a model that would require little to no fine-tuning for other similar domains [7]. Comparing the results of these two experiments allows us to determine directly if models trained on low-quality but high-quantity images from another dataset can carry over to a high-quality but low-quantity dataset, without any exposure to the latter dataset. We expect this to go poorly due to the lack of finetuning.

Finally, we attempt an experiment that pre-trains and validates on PlantVillage first, followed by finetuning on PlantLeaves and testing on PlantLeaves. By comparing this to our experiment that does not finetune and only trains on PlantVillage, we can directly compare the result of finetuning. In addition, by comparing this experiment to the experiment that only trains on PlantLeaves, we can determine if pretraining is at all beneficial in this domain with our given approaches, or if it is in fact possibly influencing our experiments negatively due to poor starting weights.

3.4. Explanation of Our Approach to Transfer Learning

We take this opportunity to note that our approach of using low-quality, high-quantity data to train and high-quality, low-quantity data to test is often the reverse of what is found in transfer learning setups, where researchers seek to use the highest quality data available to train. We felt there were some shortcomings in this approach.

First, training on the best data may lead to a very accurate model, but will not be useful for inference, if the data for inference is lower. As a rule of thumb, it is easier to downsample and throw away information, rather than upsample and generate new information.

Secondly, the best data may not always be available, and hand-curating, labeling or funding the creation of a dataset can be 'expensive' to carry out, given the nature of the difficulty of collecting data. However, low quality data can be available in spades, as it is easier to generate and takes fewer resources in the form of time, money, space, computational power and hardware. As a result, being able to train an accurate model from low-quality data is a very impactful goal. On the flip side, it may possibly be easier to collect high-quality data when a machine learning system is put into production, as people may be willing to share their personal research or even data when they hear about an ongoing study. Given the (possibly) relative paucity of data, it could be easier and more worthwhile to collect higher-quality data, since there is less to evaluate, and one may be willing to spend more to make a test input as informative as possible.

In addition, if a model trained on low-quality data can still act on features, infer what useful features are present in high-quality data or downsample the high-quality data to match the quality of the low-quality data, this would keep test accuracy comparable to training accuracy, when the model is trained on low-quality data and tested on high-quality data. A model trained on high-quality data and testing on low-quality data is more likely to see a significant decrease in performance from training accuracy to testing error, due to using more information to train than may be at one's disposal while testing.

Finally, hardware used for IOT applications can be hyperoptimized for particular tasks, like taking high-quality images. PlantVillage has many qualities, such as having segmented or augmented images as being part of the dataset and sacrificing quality of the images (understandable, given the age of the dataset), and thus, if we use it to train for now while it is the largest dataset available at our disposal, we should be cognizant of the fact that we are more likely to get higher-quality data to test our model with, rather than similar-quality data.

In general, being able to train on low-quality data may be useful for bootstrapping a model and deploying it to be tested on high-quality data; when the high-quality data is eventually labeled, a model trained on it can then replace the original model, but this initial model allows us a prototype to get access to the high-quality data in the first place, and is thus, extremely useful.

4. Dataset and Features

4.1. PlantVillage Dataset

For our low-quality leaf dataset, we utilized the New Plant Disease Classification dataset [8], also known as PlantVillage. This dataset consists of over 87,000 color images of crop leaves (both healthy and diseased).

The PlantVillage dataset is subdivided into directories for 14 different crops (namely apple, corn, cherry, blueberry, grape, peach, potato, raspberry, bell pepper, squash, soybean, strawberry, tomato, and orange). Within each crop type, the PlantVillage dataset is further subdivided into healthy and diseased examples. Diseased examples are further divided by disease (for example, for apple, there are apple scab, black rot, and cedar apple rust diseased classes). In summary, this gives us a total of 38 individual classes.

Figure 4. Healthy Corn Leaf From PlantVillage dataset

Above are some examples (Figures 4, 5) of images from the PlantVillage dataset. Note that while the quality of the images is passable, the resolution is quite poor (especially in relation to modern photographic technology). It is also important to note that the PlantVillage dataset contains many augmented images (duplicates of another image within the dataset that are rotated, flipped and cropped and zoomed)

Figure 5. Diseased Apple Leaf From PlantVillage dataset

and not in ways that are necessarily beneficial for our training as seen in the augmentations used in [10], as cropping and zooming changes the resolution of the image.

For our experiments, we wanted to repurpose the PlantVillage dataset and transform it into a dataset with binary labels - healthy or diseased. By diving into the data, we realized there was a class imbalance, as we have 26 diseased classes and 12 healthy classes.

In order to make the number of healthy and diseased images balanced, we realized we should sample more images from the healthy classes and fewer from diseased classes. Thus, we decided to sample evenly between diseased classes, as well as sample evenly between healthy classes.

We recognized that we have too many images to efficiently train over, so we decided to subsample at a rate of 10%. When there were enough original images for a class (true for all but three classes), we eliminated the augmented images and chose from what was left. Thus, we picked 390 images from each healthy class and 180 images from each diseased class. We then divided them into the train, validation and test sets using a 60/20/20 split.

4.2. PlantLeaves Dataset

Given that our experiments were focused on transfer learning, we also utilized a high-quality leaf dataset, called PlantLeaves [3]. It consisted of 4,502 images from 12 plants, with almost all of them having a set of healthy images and set of diseased images. There were 11 sets of healthy images and 11 sets of diseased images.

Figure 6. Healthy Chinar Leaf From PlantLeaves dataset

Figure 7. Diseased Chinar Leaf From PlantLeaves dataset

Above are examples of images (Figures 6, 7) from the PlantLeaves dataset. Note the high-quality resolution of both images.

Since we defined the problem as attempting to improve performance on sparse high-quality images from plentiful low-quantity images using transfer learning, we fixed the size of our test dataset from PlantLeaves to be approximately a third of that of the PlantVillage dataset. Finally, we also extracted training and validation data from the PlantLeaves dataset in order to ensure a 60/20/20 split. As a result, we ended up with the following dataset splits as seen in Table 1:

dataset	Train	Val	Test
PlantVillage (diseased)	2808	936	936
PlantVillage (healthy)	2808	936	936
PlantVillage (total)	5616	1872	1872
PlantLeaves (diseased)	999	333	333
PlantLeaves (healthy)	1002	334	334
PlantLeaves (total)	2001	667	667

Table 1. Number of samples in PlantVillage and PlantLeaves splits

Finally, in terms of data augmentation, we randomly did zooms, shears, and height/width shifts on the training data with probability .2.

5. Experiments, Results, Discussion

As detailed above, our overall guiding question for our experiments was "How effective will pretraining on low-quality leaf images (which are more common) be for inference on higher-quality leaf images (which are less common but more accurately represent inference conditions)?" Our primary metric for evaluating this was accuracy.

5.1. Hyperparameter Tuning

In terms of hyperparameter tuning, we experimented with a variety of learning rates. We experimented with three learning rates: $\alpha = 10^{-3}, 10^{-4}, 10^{-5}$. We found that while training was quite slow with learning rates of 10^{-4} and 10^{-5}, it progressed at a reasonable pace at $\alpha = 10^{-3}$. We also experimented with a batch sizes of $64, 128, 256$ and found that 128 worked best. Finally, as detailed above,

we used the Keras Stochastic Gradient Descent (SGD) optimizer with a momentum rate of .9.

5.2. Experiments

We split our experiments into two phases. First, we tested the effectiveness of our custom CNN architecture. We did this by training our baseline and custom architecture on the PlantVillage dataset and evaluating each architecture's performance. For the second phase, we tested out the effectiveness of a transfer learning approach.

5.2.1 Training and Testing on PlantVillage with a Simple Architecture

For this experiment, we used our 'simple' CNN architecture that had only two convolutional layers and only two fully connected layers, but many parameters. As mentioned earlier, we seek to test different architectures to check if a shallower model with more parameters could outperform a deeper model with fewer parameters.

We tested this architecture on PlantVillage itself just to ensure the model performed well on the primary dataset that it was trained on.

5.2.2 Training and Testing on PlantVillage with a Custom Architecture

For this experiment, we used our custom CNN architecture that had three convolutional layers, utilized BatchNorm and ReLU, and three fully connected layers, but far fewer overall parameters at less than a third of the size of the 'simpler' CNN architecture.

We also tested this architecture on PlantVillage itself just to ensure the model performed well on the primary dataset that it was trained on.

5.2.3 Training and Testing on PlantLeaves with a Custom Architecture

For this experiment, we continued using our custom CNN architecture, since it performed well on PlantVillage, and sought to understand how training and testing on the PlantLeaves dataset would impact performance so that we could later contrast it to models trained or pre-trained on PlantVillage.

5.2.4 Training on PlantVillage and Testing on PlantLeaves

With our previous attempts in mind, we pre-trained a model on the lower-quality PlantVillage dataset, then directly tested on the higher-quality PlantLeaves dataset to see how well inference would work without any finetuning on the higher-quality PlantLeaves dataset.

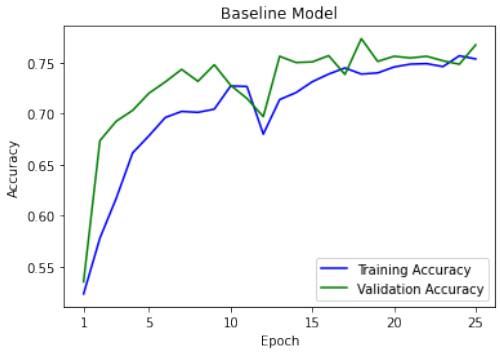

Figure 8. Training and Validation Accuracy across 25 epochs for the Baseline Model on the PlantVillage dataset

5.2.5 Training on PlantVillage, Finetuning and Testing on PlantLeaves

Finally, we pre-trained a model on the lower-quality PlantVillage dataset, then fine-tuned it on the higher-quality PlantLeaves dataset, before testing on that dataset as well. The purpose of this was to see how much finetuning on PlantLeaves would improve the model as we expect this process to improve embeddings and generate them in a manner that would make them more readily applicable to the PlantLeaves dataset. In doing so, it served as a reference to see how much of a benefit or detriment looking at a lower-quality dataset would be for inference on a higher-quality dataset.

5.3. Results

5.3.1 Training and Testing on PlantVillage with a Simple Architecture

We trained a simple model on solely the PlantLeaves dataset, and continued for 25 epochs. Both the training and validation accuracy increased in parallel, though they were ultimately plateauing after 25 epochs, and reaching a peak accuracy of around 75%. The graph, which depicts validation accuracy being even higher than training accuracy for many epochs, can be seen in Figure 8. We observe a test accuracy of 78.03% on the PlantVillage dataset.

5.3.2 Training and Testing on PlantVillage with a Custom Architecture

We trained our custom model on solely the PlantLeaves dataset, and continued for 25 epochs. Both the training and validation accuracy increased over time, though the training accuracy mainly just increased with few dips, while the validation accuracy was more volatile, though continuing an upwards trend. We were surprised by this volatility given that the learning rate was relatively small, at 1e-3. Again,

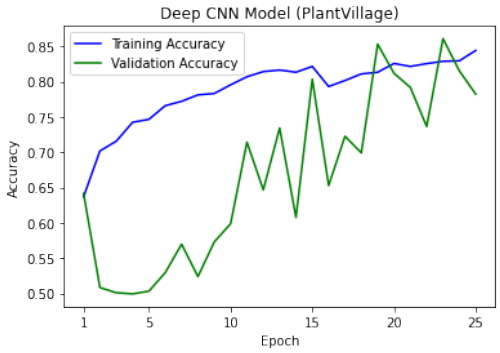

Figure 9. Training and Validation Accuracy across 25 epochs for the Deep CNN Model on the PlantVillage dataset

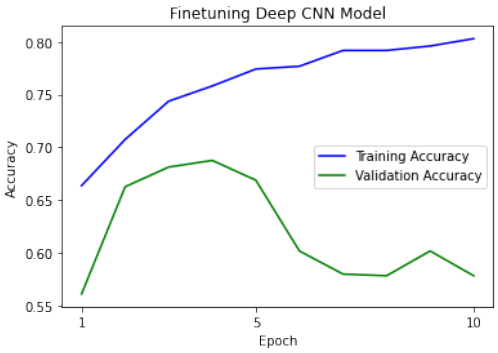

Figure 10. Training and Validation Accuracy across 10 epochs for finetuning the Deep CNN Model on PlantLeaves. This model was pretrained on PlantVillage for 25 epochs.

both the training and validation accuracy were ultimately plateauing after 25 epochs, though they reaching a peak accuracy of around 85%, far higher than what was seen in the simpler architecture. The graph can be seen in Figure 9. We observe a test accuracy of 86.21% on the PlantVillage dataset.

5.3.3 Training and Testing on PlantLeaves with a Custom Architecture

We trained our custom model on solely the PlantLeaves dataset for 25 epochs. We observed a test accuracy of 53.22% on the PlantLeaves dataset.

5.3.4 Training on PlantVillage and Testing on PlantLeaves

We continue using the model trained in Section 5.2.2 with the results seen in Figure 9. We observe a test accuracy of 50.82% on the PlantLeaves dataset.

5.3.5 Training on PlantVillage, Finetuning and Testing on PlantLeaves

We continue using the model trained in Section 5.2.2 with the results seen in Figure 9. We observe a test accuracy of 52.77% on the PlantLeaves dataset. We recognize that this in fact, does demonstrate that finetuning works to improve our accuracy, as we increase our test accuracy by almost 2% with just 10 epochs of training.

5.4. Summary of Results

Our initial experiments helped us to determine which architecture was more suited for this problem domain. Since our custom architecture far outperformed our baseline architecture by over 8%, as seen in Table 2, we continued to use the custom architecture for the rest of our experiments.

Model	Test Accuracy on Plant Village
Baseline	78.03 %
Custom	86.21 %

Table 2. Comparing our baseline and custom models.

We see in Table 3 that no model does much better than 50% accuracy on the PlantLeaves test set, including the model specifically trained on PlantLeaves, though that model does perform the best overall, with a 53.22% accuracy.

Model	Test Accuracy
PlantLeaves Train	53.22 %
Only Pretrain	50.82 %
Pretrain + Finetune	52.77 %

Table 3. Comparing performance on the PlantLeaves test set.

5.5. Discussion

First, we were successful in creating a lightweight custom deep CNN that outperformed a shallower baseline CNN by over 8%. This is impressive given the fact that our custom CNN had about a third of the parameters of the baseline model. This is extremely important especially in the context of on-edge inference.

Secondly, we see that no model did exceptionally well on the PlantLeaves test dataset. This was regardless of what it was trained on or if it was pre-trained and fine-tuned. Given that the dataset was pretty even split between the healthy and diseased labels, guessing a random label would achieve an approximately 50% accuracy.

While our transfer learning approach did not work well, it's important to note that the model that trained exclusively on the PlantLeaves dataset only achieved a test set accuracy of 53.22%. This suggests that this is an extremely different problem requiring more data, a different architecture, a different combination of hyperparameters and perhaps more

training time to solve.

We were surprised that our accuracy was so low for the model that pre-trained on PlantVillage and then was fine-tuned on PlantLeaves, ultimately giving it the most training time and data. In order to better understand our results, we generated a saliency map for one of the input images we had wrong. In the input image in Figure 11, we have a healthy Chinar leaf. However, this is incorrectly classified as diseased. We can see from the generated saliency map in Figure 11 that the background, not the leaf itself, appears in white and is being used to classify the leaf. In the original PlantVillage dataset, we see that pictures are generally taken across a lighter background, and darker colors generally indicate disease on a plant leaf. Thus, given the background of the misclassified image, it suggests that our model used the darker background color to classify it as diseased, when it was in fact, healthy. This suggests that our model approach to this problem may need to change, as we may need to either brighten the picture to make the background appear lighter, or first try image segmentation, before attempting to classify the image.

In addition, many of the images in the PlantVillage dataset were already segmented and completely omitted from the image. Given that this was not the case in PlantLeaves, the domain of the problem in the PlantLeaves dataset may have been too different for transfer learning to succeed, without preprocessing the images in Plant Leaves.

Figure 11. A saliency map showing how a model pre-trained on PlantVillage makes an error.

6. Conclusion

We found that it was possible to create a lightweight deep CNN that could outperform a baseline model with far more parameters. The results are in-line with work from many other application areas, showing the power of going deeper with our model architectures. We also found that transfer learning on low-quality plant images did not seem useful for this task. With that said, given the poor results of even a model trained on PlantLeaves on the test set of PlantLeaves, we believe that there may have been issues around requiring more data or training time.

6.1. Future Work

First, we would attempt to rerun our approaches with more data, as we believe that we may have suffered from having too few datapoints.

The performance of our model pretrained on PlantVillage and then fine-tuned on PlantLeaves was disappointing. In part, we believe that this may be owed to the fact that we tried to group all healthy leaves together into one class and all diseased leaves together into another. We would like to modify the approach of [4] and apply it towards our own work. We could this by first pretraining a model on all 38 classes of PlantVillage, which we hope would generate effective embeddings from the first several layers, while the last layer would be used to more specifically classify which of the 38 classes a leaf belonged to. Then, we could remove the last layer of the model and replace it with a layer with 22 nodes, such that it could be applied to PlantLeaves. Then, we could finetune on PlantLeaves, which would allow for modification of the embeddings we already have pretrained the model to generate with PlantVillage and train the final layer to utilize the embeddings for classification.

We would also like to test how modifying the architecture with more complex blocks such as 'squeeze-and-excitation' networks as seen in [10] would impact the level of data needed to achieve high-quality results.

7. Contributions, Acknowledgements

We would like to thank Professors Fei-Fei Li, Jiajun Wu, Ruohan Gao and the TAs for their time and expertise and wonderful experience throughout this class. In addition, we would like to thank our mentor Haochen Shi, for his time and feedback.

References

[1] A. F. Aji, N. Bogoychev, K. Heafield, and R. Sennrich. In neural machine translation, what does transfer learning trans-

fer? Association for Computational Linguistics, 2020.

[2] J. Boulent, S. Foucher, J. Théau, and P.-L. St-Charles. Convolutional neural networks for the automatic identification of plant diseases. *Frontiers in Plant Science*, 10, 2019.

[3] A. S. U. P. CHOUHAN, Siddharth Singh; Kaul. A database of leaf images: Practice towards plant conservation with plant pathology. `https://data.mendeley.com/datasets/hb74ynkjcn/1`, 2019.

[4] M. S. Elmahdy, S. S. Abdeldayem, and I. A. Yassine. Low quality dermal image classification using transfer learning. In *2017 IEEE EMBS International Conference on Biomedical Health Informatics (BHI)*, pages 373–376, 2017.

[5] J. Graziano Da Silva. Feeding the world sustainably.

[6] S. Gupta, A. Chopade, N. Jain, and A. Bhonde. Farmer's assistant: A machine learning based application for agricultural solutions, 2022.

[7] X. Li, X. Long, Y. Xia, and S. Li. Low resource style transfer via domain adaptive meta learning, 2022.

[8] S. Mohanty. Plantvillage-dataset. `https://github.com/spMohanty/PlantVillage-Dataset`, 2018.

[9] S. P. Mohanty, D. P. Hughes, and M. Salathé. Using deep learning for image-based plant disease detection. *Frontiers in Plant Science*, 7, 2016.

[10] C. Qiu, S. Zhang, C. Wang, Z. Yu, H. Zheng, and B. Zheng. Improving transfer learning and squeeze- and-excitation networks for small-scale fine-grained fish image classification. *IEEE Access*, 6:78503–78512, 2018.

[11] A. Ramezani-Kebrya, A. Khisti, and B. Liang. On the generalization of stochastic gradient descent with momentum, 2018.

[12] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, and Q. He. A comprehensive survey on transfer learning, 2019.