Leveraging Transfer Learning and Knowledge Distillation for Leaf Disease Classification

David Smith
CSE and Robotics Department
University of Michigan
Ann Arbor, Michigan
Email: smitd@umich.edu

Jess Wu
CSE and Robotics Department
University of Michigan
Ann Arbor, Michigan
Email: jessyw@umich.edu

William Hasey
Robotics Department
University of Michigan
Ann Arbor, Michigan
Email: whasey@umich.edu

***Abstract*—Identification of diseased crops through the symptoms displayed on their leaves can help farmers manage large-scale operations. Integrating deep learning with robotics provides a promising way for farmers to maintain their crops in an efficient and cost-effective manner. However, deep learning models are often too large and computationally complex to be deployed in agricultural robots. Previous research has been conducted on small datasets and has produced models that can identify diseases for a specific species of crop. Drawing from previous research on the study of apple leaves and associated diseases, along with studies on knowledge distillation, this work proposes a lightweight model that can classify 17 common diseases spanning 14 different crop species.**

I. Introduction

Agricultural robotics is a fast-growing area of research—in particular, many researchers are focusing on using automation to perform tedious everyday tasks such as weeding, scouting, and harvesting [1]. Properly trained robots would be less prone to missing important information when working fields, and would not fall victim to dangerous environmental factors such as extreme heat, pesticides, and hazardous machinery. As robotics has become a more prominent field in engineering, interest in agricultural robotics has spiked [2], and deep learning has been a heavily researched area relating to this topic. However, as interest in this specific area of robotics has increased, the requirements for a practical machine learning model have become narrower. Any model created must be highly accurate, generalizable to real-world data, and require as little memory and computational power as possible, such that a robot's ability to perform tasks remains more efficient than a human's.

Therefore, the purpose of this work is to develop a lightweight model that expands on the capabilities of previous models while maintaining a high enough accuracy to be a practical solution in the world of agricultural robotics. Using deep transfer learning, we train the large, powerful model DiseasedCNN, aiming to maximize accuracy over all else. Then, leveraging the advantages of knowledge distillation, we propose DiseasedCNN-Lite: a lightweight model that adopts DiseasedCNN's accuracy, while being a fraction of the size.

II. Related Works

Image processing and classification of diseased leaves has been researched before, and a majority of studies are specific to one crop (apples and tomatoes are by far the most common). Many different machine learning architectures have been attempted—some deep models, some not—and most have been able to achieve test accuracies upwards of 90%-95% [3] [4] [5] [6] [7] [8] [9]. These studies focused on one crop species and had relatively small datasets of a few thousand training images creating a lack of versatility.

A. DeepCNN

DeepCNN is a convolutional neural network built by [3] that can identify three different apple leaf diseases, as well as a healthy leaf. This study laid the foundation for our work on disease detection. After training for 1000 epochs, DeepCNN achieved an accuracy of 98% on a dataset consisting of 3,171 images (2,228 training, 634 validation, 319 testing). To lay the foundation for our model, we attempted to reproduce the final results of [3]. The study provided the dataset used and outlined the exact architecture of DeepCNN, including specific dimensions for each layer; however, specifics for the data augmentation performed were not provided.

In our attempt to reproduce these results, we discovered that DeepCNN would drastically overfit to the data when trained for 1,000 epochs. Instead, we achieved 96% testing accuracy after 250 epochs of training, and training for any longer would lead to a decrease in overall model performance. We theorize that this discrepancy could be a result of the differing data augmentation. Since the operations performed on the images were not detailed by [3], we researched and used relatively common operations (horizontal and vertical flips, shearing, scaling, shifting) and applied them to each training image with a 40% probability. Data augmentation is used to prevent overfitting, but its misuse can harm model performance and actually increase the chance of overfitting. Therefore, we believe that our data augmentation was substantially different, and led to the drastic change in accuracy as the number of training epochs passed 250.

III. Design Considerations

A. Dataset

One limiting factor of the DeepCNN network proposed by [3] was that the dataset used only contained four classes on a single species. Therefore, we first aimed to build a network that could identify many more species and diseases at a similar rate. We used the Plant Village dataset [10]. This dataset contains 54,305 images spanning 14 different species of crop and 17 common diseases, amounting to 38 separate classes. We split the dataset into training, testing, and validation using the standard 70-20-10 split.

As a preemptive measure to prevent overfitting, we probabilistically applied different augmentation operations to each training image. Given that there were only approximately 1,000 images per class in the Plant Village training set, we determined that data augmentation would be our most effective tool against overfitting. Our data augmentation consisted of shearing, scaling, horizontal and vertical flipping, and elastic distortion.

B. Backbone

To determine which pretrained model to use as a backbone for transfer learning, we benchmarked the performance of seven commonly used models against the Plant Village dataset [10]. This process included adding one fully connected layer to the end of each pretrained model, and then training for 20 epochs. As seen in Figures 1 and 2, EfficientNet-B0 and ResNet50 performed best. Both reached a loss below 3.0 and a validation accuracy above 80%. After this benchmarking process, we trained DiseasedCNN for three epochs with both EfficientNet-B0 and ResNet50 to determine which model meshed best with our fine-tuning layers. We determined that ResNet50's feature extraction was better suited for our needs, as EfficientNet-B0 achieved a test accuracy of 61.51% and ResNet50 achieved a test accuracy of nearly 75%. Therefore, we decided to use ResNet50 as our pretrained backbone for DiseasedCNN.

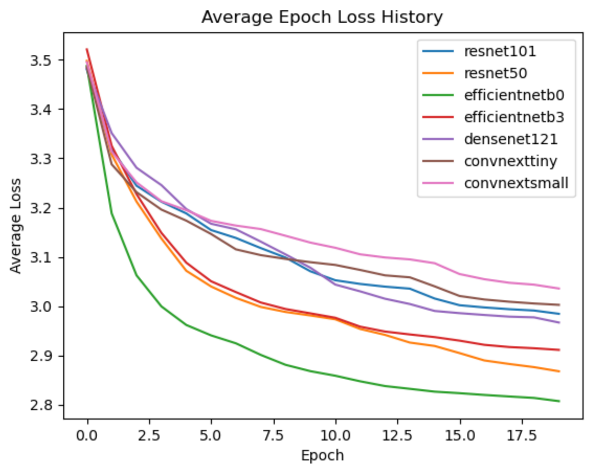

Fig. 1. Loss per epoch for each pretrained backbone candidate

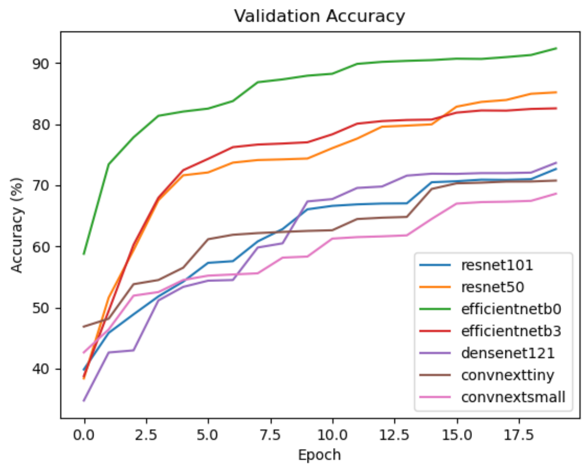

Fig. 2. Validation accuracy per epoch for each pretrained backbone candidate

IV. DiseasedCNN

DiseasedCNN is the powerful model that we built to serve as a teacher during the knowledge distillation process. It relies on ResNet50's feature extraction to provide a basis for its classification abilities, and utilizes fine-tuning layers to calibrate its prediction to our dataset. It also employs a rectified linear unit (ReLU) activation function to provide nonlinearity, and softmax is used on the output to produce a probabilistic prediction.

A. Structure

The foundation of DiseasedCNN is a ResNet50 backbone with the last three layers (adaptive average pooling, flattening, and fully connected) removed. This allows us to feed the output of the network into our fine-tuning layers, while maintaining the spatial information that the network has extracted.

The fine-tuning layer architecture is as follows: two convolutional layers with ReLU activations on the outputs, one batchnorm layer and one adaptive average pooling layer, and two fully connected layers, with ReLU activation after the first layer and softmax after the second. Table 1 provides specifics on DiseasedCNN's layer architecture.

B. Training

DiseasedCNN trained for 50 epochs on the Plant Village [10] dataset, with model validation performed every 13 batches. During training, we used an Adam optimizer with a learning rate of 0.0005, a β_1 value of 0.9, and a β_2 value of 0.999. In addition, we used an exponential learning rate scheduler with a γ factor of 0.9. Our loss function for DiseasedCNN's training was the standard PyTorch cross-entropy loss function. The model was trained for approximately two hours, and while the average epoch loss remained relatively high (above 2.5, shown in Figure 3), the model reached a final validation accuracy of 93%, as seen in Figure 4.

TABLE I

Layer	Input Dimension	Output Dimension	Stride	Padding
ResNet50	3 x 256 x 256	2048 x 8 x 8	1	1
Conv + ReLU	2048 x 8 x 8	512 x 8 x 8	1	1
Conv + Batchnorm + ReLU	512 x 8 x 8	256 x 8 x 8	1	1
Adaptive Average Pool	256 x 8 x 8	256 x 1 x 1	-	-
Flatten	256 x 1 x 1	256	-	-
Fully Connected + ReLU	256	64	-	-
Fully Connected + Softmax	64	38	-	-

C. *Capabilities*

Ultimately, DiseasedCNN achieved a test accuracy of 93%, with an average inference time of 1.15 milliseconds on an RTX 4090 GPU. We expect this inference time to increase by a factor of roughly 100 when running on an average CPU, which would be approximately 115 milliseconds. Additionally, DiseasedCNN is 140 megabytes in size.

D. *Discussion*

DiseasedCNN's inference time is relatively high compared to other models of the same size. We would like to see this value within the range of 10 to 30 milliseconds. This is particularly important for any integration with agricultural robotics, because a longer inference time means that it takes more energy and computational power to run inference. Maximizing battery life is a major aspect of agricultural robotics. Furthermore, DiseasedCNN is too large for many agricultural robots and drones. This is largely due to its ResNet50 backbone, which provides the necessary feature extraction and enables such high classification accuracy. Overall, DiseasedCNN is a good start, but in order to be a practical solution in the field of optimized agricultural robotics, DiseasedCNN would need to see a drastic reduction in size and inference time.

V. DiseasedCNN-Lite

DiseasedCNN-Lite is a much smaller model that we built to serve as a student during the knowledge distillation process. Instead of relying on a large, pretrained backbone such as ResNet50 to perform feature extraction, it extracts a small number of foundational features from a given image, while focusing on learning to predict the output of its teacher, DiseasedCNN. Since this model is much smaller than DiseasedCNN, we opted to use a Leaky ReLU activation function to prevent any gradients from vanishing during the training process.

A. *Structure*

DiseasedCNN-Lite consists of three convolutional layers with Leaky ReLU activations on the outputs and pooling layers following the activation function. The first two pooling layers are max pooling with a 2×2 kernel, and the third is an adaptive average pooling layer that shrinks the spatial features to 1×1. Afterwards, the output is flattened and passed through one final fully connected layer.

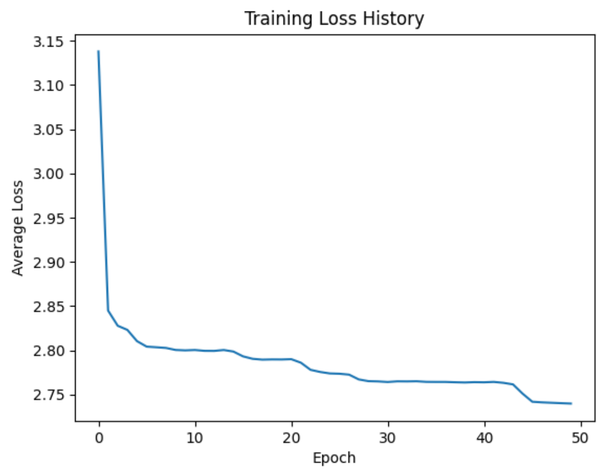

Fig. 3. DiseasedCNN average training loss per epoch

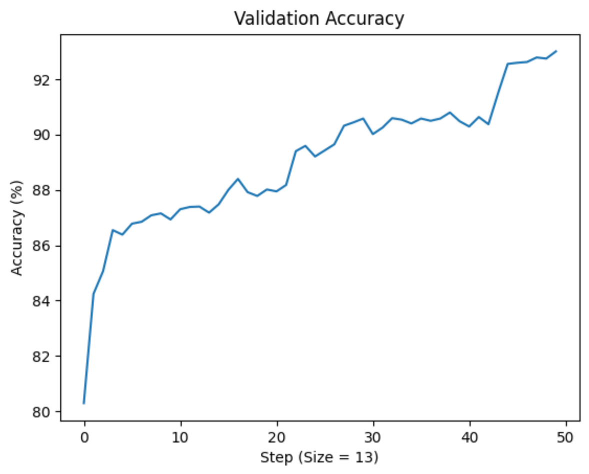

Fig. 4. DiseasedCNN validation accuracy, measured every 13 batches

B. *Distillation Loss Definition*

The process of knowledge distillation can be implemented in many different ways. Feature maps, weights, biases, or any other aspect of the student and teacher models can be compared at any point in the training process, and used to calibrate the student. For our implementation, we opted to calculate the student model's loss with a weighted combination of the student's loss and the teacher's loss. A defining characteristic of this process is the temperature value. This value defines the smoothness of the probability distributions produced by the teacher. A higher temperature leads to a less concentrated distribution, whereas a lower temperature leads to a more peaked distribution. This dictates how much the student model will focus on the most confident predictions made by the teacher.

First, we define p_i and q_i as the result of the softmax with temperature function, where T is the temperature value, and where T_i and S_i are the teacher and student output logits, respectively.

$$p_i = \frac{e^{T_i/T}}{\sum_{j=1}^{C} e^{T_j/T}}, \quad q_i = \frac{e^{S_i/T}}{\sum_{j=1}^{C} e^{S_j/T}}$$

Next, we use the Kullback–Leibler divergence to measure the difference in the probability distributions predicted by the student and the teacher. This enables the student to learn the probability distribution produced by the teacher for a given input image, and allows for a more well-rounded prediction than could be achieved by purely basing its loss on the discrete (correct or incorrect) output of cross-entropy loss.

$$KL(P||Q) = \frac{1}{N}\sum_{n=1}^{N}\sum_{i=1}^{C} p_i^{(n)} \log\left(\frac{p_i^{(n)}}{q_i^{(n)}}\right)$$

Then, we define soft loss and hard loss. Soft loss is defined as the result of the Kullback-Leibler divergence, multiplied by a scale factor T^2. We introduce this T^2 term to correct for the scaling that occurs due to the temperature term within the softmax function. Hard loss is defined as the standard cross-entropy loss function.

$$\mathcal{L}_{\text{soft}} = T^2 \cdot \frac{1}{N}\sum_{n=1}^{N}\sum_{i=1}^{C} p_i^{(n)} \log\left(\frac{p_i^{(n)}}{q_i^{(n)}}\right)$$

$$\mathcal{L}_{\text{hard}} = -\sum_{i=1}^{C} p_i \log(q_i)$$

Lastly, we produce the final loss of the student model by performing a weighted sum of the hard loss and the soft loss. The weighting factor α is a hyperparameter tuned during training, with the constraint that the sum between α and $1-\alpha$ must equal 1.

$$\mathcal{L} = (\alpha) \cdot \mathcal{L}_{\text{hard}} + (1-\alpha) \cdot \mathcal{L}_{\text{soft}}$$

C. Training

DiseasedCNN-Lite trained for 100 epochs on the Plant Village dataset [10], with model validation performed at the end of each epoch. During training, we used many of the same hyperparameters as DiseasedCNN. We used an Adam optimizer with a learning rate of 0.0005, a β_1 value of 0.9, and a β_2 value of 0.999. We also used a learning rate scheduler to help optimize the training of the student model, with a γ factor of 0.9. However, for DiseasedCNN-Lite we added a weight decay factor of 0.00001 to avoid large weights and prevent overfitting. Furthermore, DiseasedCNN-Lite trained with a T value of 4, as we chose to soften the teacher model's probability distribution, and an α value of 0.7, putting more emphasis on hard loss than soft loss. Table 2 details the hyperparameters used for training both DiseasedCNN and DiseasedCNN-Lite.

TABLE II
TRAINING HYPERPARAMETERS

	DiseasedCNN	DiseasedCNN-Lite
Learning Rate	0.0005	0.0005
Weight Decay	-	0.00001
γ	0.9	0.9
T	-	4
α	-	0.7
β_1	0.9	0.9
β_2	0.999	0.999
Epochs	50	100

The model trained for approximately one and a half hours. As seen in Figure 6, the average epoch loss started at an unusually high value of 16. We believe that this stems from the model attempting to integrate the teacher model's probability distribution into its loss calculation. The use of a high T value meant that the teacher's probability distribution did not provide as clear of a choice for DiseasedCNN-Lite as it could have, but we believed this paired well with an α value that put more emphasis on DiseasedCNN-Lite's loss. Our choice of hyperparameters was tailored towards using the teacher model to gently guide DiseasedCNN-Lite toward the correct prediction, rather than forcing DiseasedCNN-Lite to accept the teacher's prediction and potentially harm its ability to generalize to real-world data.

DiseasedCNN-Lite also began training with a very low validation accuracy compared to its teacher, DiseasedCNN. However, this was expected, as there was no feature extraction performed by a pretrained backbone. As seen in Figure 7, the validation accuracy quickly rose to 70% in 20 epochs, and then began to rise much more slowly over the next 80 epochs.

D. Capabilities

DiseasedCNN-Lite achieved a test accuracy of 87%, with an average inference time of 0.007 milliseconds on an RTX 4090 GPU. As with DiseasedCNN, we expect this inference time to increase by a factor of roughly 100 when running on an average CPU, which would be approximately 0.7 milliseconds. Overall, DiseasedCNN-Lite was 390 kilobytes in size.

E. Discussion

DiseasedCNN-Lite's inference time was much faster than DiseasedCNN's, and will likely require significantly less energy and computational power to run on an agricultural robot.

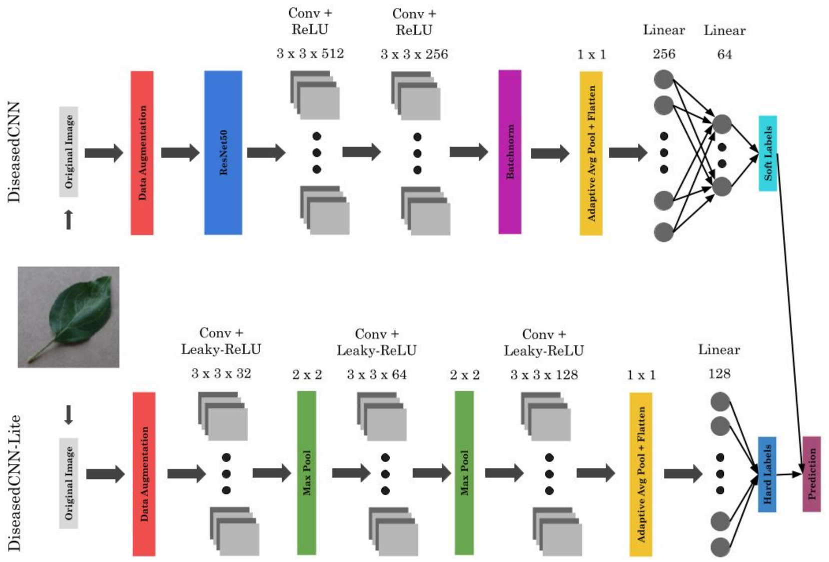

Fig. 5. Knowledge Distillation Model

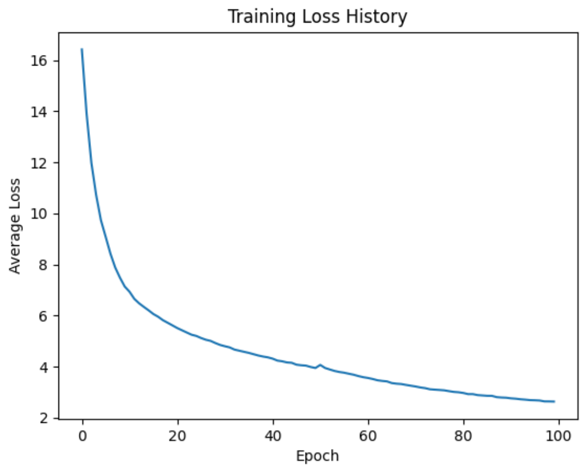

Fig. 6. DiseasedCNN-Lite average training loss per epoch

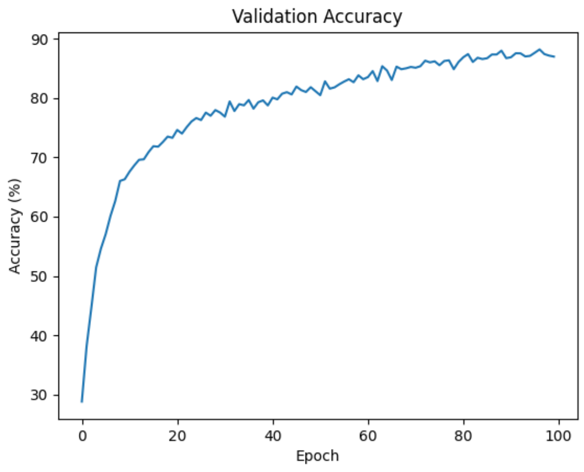

Fig. 7. DiseasedCNN-Lite validation accuracy per epoch

In addition to this, DiseasedCNN-Lite was 334 times smaller than its teacher, DiseasedCNN. The total size of the model, along with the shortened inference time, makes DiseasedCNN-Lite a more practical solution than DiseasedCNN, and opens the door to integration with agricultural robots and drones. Although a test accuracy of 87% is relatively low compared to other existing models (which routinely achieve accuracies above 90%), DiseasedCNN-Lite provides a strong foundation for future research into training smaller student models.

We believe that with slight changes to hyperparameters, and many more epochs of training, DiseasedCNN-Lite could increase its capabilities. We would have to be careful not to overfit to our dataset, and to prevent this our best option would likely be to increase our data augmentation. To best support an increase in DiseasedCNN-Lite's abilities, we would continue to calibrate DiseasedCNN in order to optimize its accuracy. Although unlikely, it is possible that DiseasedCNN-Lite could achieve a higher test accuracy than its teacher, DiseasedCNN. However, a more accurate teacher model would only be more helpful during training.

VI. Conclusion

DiseasedCNN-Lite achieved a respectable accuracy of 87%, while being an extremely lightweight model totaling only 390 kilobytes in size. Compared to the capabilities of its teacher, DiseasedCNN-Lite was capable of similar accuracies while shrinking the size of the model by a factor of 334. Overall, this model is still not ready to be deployed and tested with agricultural robots and drones, but it lays a solid foundation for future work.

One of the main limitations of DiseasedCNN-Lite is that it can only predict diseases given an image containing only one leaf. A potential solution to this problem is to use a vision transformer as a pretrained backbone, and teach DiseasedCNN-Lite to not only identify diseased leaves, but also locate leaves with bounding boxes if given an image with multiple leaves. We would also like to increase the test accuracy of DiseasedCNN-Lite, and perform tests on how well it generalizes to real-world data. The next step after those improvements is to deploy the model onto an agricultural robot or drone, and test its capabilities in the real world.

Acknowledgments

A special thanks to Xiaoxiao Du and the rest of the ROB 498 Staff.

References

[1] R. Shamshiri, C. Weltzien, I. Hameed, I. Yule, T. Grift, S. Balasundram, L. Pitonakova, D. Ahmad, and G. Chowdhary, "Research and development in agricultural robotics: A perspective of digital farming," *International Journal of Agricultural and Biological Engineering*, vol. 11, pp. 1–14, 2018.

[2] S. Hajjaj and K. Sahari, "Review of agriculture robotics: Practicality and feasibility," *IEEE International Symposium on Robotics and Intelligent Sensors*, vol. 4, pp. 194–198, 2017.

[3] V. K. Vishnoi, K. Kumar, B. Kumar, S. Mohan, and A. A. Khan, "Detection of apple plant diseases using leaf images through convolutional neural network," *IEEE Access*, vol. 11, pp. 6594–6609, 2022.

[4] M. Agarwal, S. Gupta, and K. Biswas, "A new conv2d model with modified relu activation function for identification of disease type and severity in cucumber plant," *Sustainable Computing: Informatics and Systems*, vol. 30, 2021.

[5] P. Wspanialy and M. Moussa, "A detection and severity estimation system for generic diseases of tomato greenhouse plants," *Computers and Electronics in Agriculture*, vol. 178, 2020.

[6] P. Jiang, Y. Chen, B. Liu, D. He, and C. Liang, "Real-time detection of apple leaf diseases using deep learning approach based on improved convolutional neural networks," *IEEE Access*, vol. 7, 2019.

[7] S. Kodors, G. Lacis, O. Sokolova, V. Zhukovs, I. Apeinans, and T. Bartulsons, "Apple scab detection using cnn and transfer learning," *Agronomy Research*, vol. 19, pp. 507–519, 2021.

[8] Z. urRehman, M. Khan, F. Ahmed, R. Damaševičius, S. Naqvi, W. Nisar, and K. Javed, "Recognizing apple leaf diseases using a novel parallel real-time processing framework based on mask rcnn and transfer learning: An application for smart agriculture," *IET Image Process*, vol. 15, p. 2157–2168, 2021.

[9] C. B. amd Jiamin Wang, Y. Duan, B. Fu, J.-R. Kang, and Y. Shi, "Mobilenet based apple leaf diseases identification," *Mobile Networks and Applications*, vol. 27, pp. 172–180, 2020.

[10] M. Singh, "Plant village dataset," tech. rep., Kaggle, 2021.