Vineyard Vigilance: Harnessing Deep Learning for Grapevine Disease Detection

Rajarshi Mandal

9/30/23
Inspirit AI Research Project

Abstract

The Food and Agriculture Organization of the United Nations (FAO) estimates that 20 to 40 percent of crops produced around the world are lost to pests, costing $220 billion dollars annually. With a high nutritional and medicinal value, grapes are one of the fruits that society depends on. Most industrialized grapevine orchards currently rely on human vision for disease detection, which causes a significant lag in the tracking of grapevine diseases. This results in poor crop yields and fruit quality. Automating grapevine leaf disease detection using machine learning will lead to sustainable farming. Deep learning, a branch of machine learning, is well-suited for learning from image data. Convolutional Neural Networks (CNN), a type of deep learning model, are employed to accurately classify healthy and disease-affected grapevine leaf images. A baseline CNN model is developed along with other CNN models, namely DenseNet121, EfficientNetB7, MobileNetV2, ResNet50, and VGG16. A max-voting ensemble that includes the most accurate CNN models is then deployed on the web through a website. Consequently, grapevine farmers and other users can detect the three common grapevine leaf diseases – black rot, Esca, and leaf blight, as well as healthy leaves by uploading their own images from the orchard.

1. Introduction

According to the International Organization of Vine and Wine, 77.8 million tons of grapes are grown annually. Grapes have been a staple of people's diets since ancient times, and are usually used for direct consumption, wine, and raisins. Unfortunately, grapevine diseases such as black rot, Esca, and leaf blight have reigned in orchards, vastly reducing crop yields and forcing farmers to overuse expensive pesticides. Black rot is a fungal disease triggered by *Guignardia bidwellii* in hot and humid weather. [1] It renders grapes inedible and has been called the “Achilles Heel” of grape production in the Middle East. Esca is a trunk disease sometimes caused by *Phaeoacremonium aleophilum* that usually manifests during July and August. [2] In France, 13% of vineyards are affected yearly by ECSA, netting a cumulative loss of over 1 billion euros. Leaf blight is a bacterial disease caused by *Xylophilus ampelinus*. Infected vineyards have reported losses of over 70% of typical yields. These diseases are problematic due to increased pesticide usage and decreased crop yields, significantly decreasing grapevine orchard profitability. Luckily, there are solutions to mitigate these diseases.

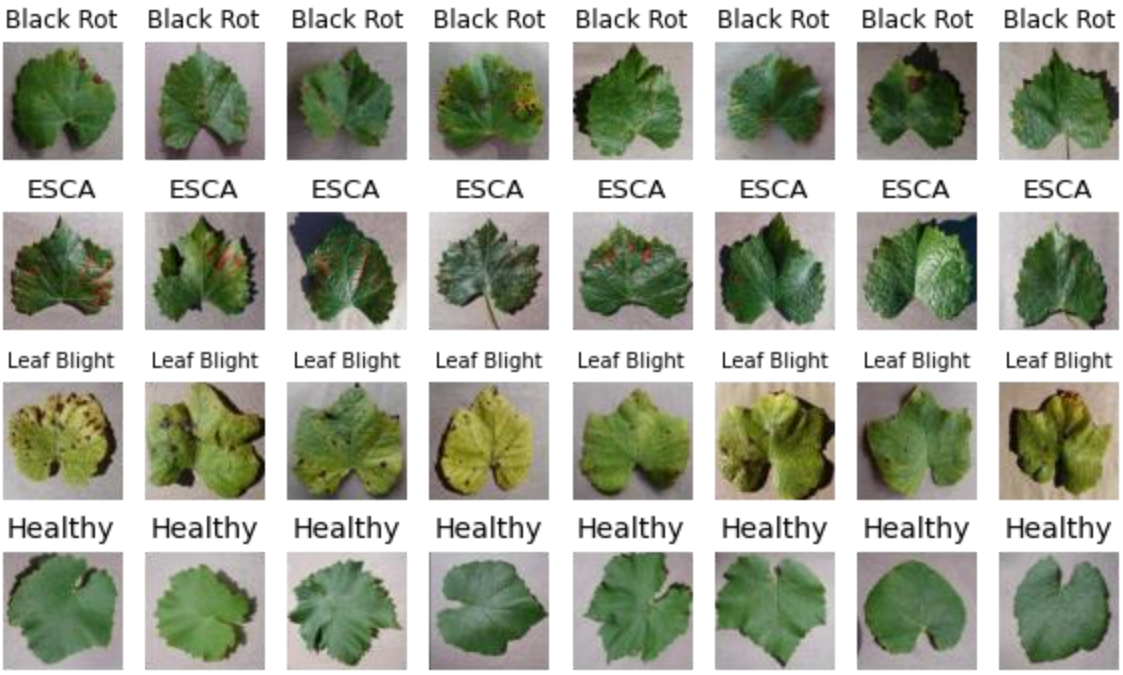

Figure 1: Images showing grapevine leaves that are infected with black rot, Esca, leaf blight, or are healthy.

Efforts have been made to reduce the spread of grapevine diseases with and without machine learning. Experts have been trained to identify and classify diseases visually. [3] However, this solution is inaccurate and economically infeasible due to visual fatigue and human error. Consequently, some researchers have tried using Support Vector Machines (SVMs) to classify grapevine diseases. [4] They fed grapevine leaf statistics to an SVM to predict occurrences of grapevine diseases. Unfortunately, these models had accuracy scores of less than 95%, rendering them unsuitable for farmers. Eventually, Convolutional Neural Networks (CNNs) were popularized, and researchers made relatively accurate models. [5] However, a novel solution could classify diseases more efficiently and accurately from lower-resolution images.

This research aims to understand which architectures work best for grapevine plant disease detection. Another goal is to create a website accessible to farmers that can predict if a grapevine plant is infected from a leaf picture. To complete these, one must collect data, perform exploratory data analysis, and augment data. Afterward, a baseline model will be trained, and transfer learning will be performed using a variety of model architectures. These models will be evaluated and compared using training and validation curves, accuracy scores, confusion matrices, classification metrics, and error visualizations. Training and validation curves help find underfitting and overfitting models. Accuracy scores provide a numerical value that summarizes performance. Error visualizations enable us to find a model's pitfalls. The final ensemble model will be used to create a website where grapevine farmers and other users can easily upload grapevine leaf images for classification by the model.

The proposed model will be significantly more robust than previous models due to data augmentation. [6] This will increase its effectiveness in a real-world setting where professional photographers are not taking pictures of grapevine leaves. In addition, most models created so far

are unavailable to farmers. The website is designed to serve as a platform that connects farmers to state-of-the-art machine learning models that predict diseases quickly and accurately.

2. Dataset

The original dataset used for this project was published by the Kaggle user "Pushpa Lama" on July 1, 2021 at 14:33:41 (Eastern Daylight Time) and was titled "Grape_disease." It contained four classes (black rot, Esca, leaf blight, and healthy) with about 2,000 training images and 500 testing images each. The images were stored as Joint Photographic Experts Group (JPG) files with a size of 256 pixels by 256 pixels. Consequently, the aspect ratio was 1:1. In total, there were 7,260 training images and 1,805 testing images.

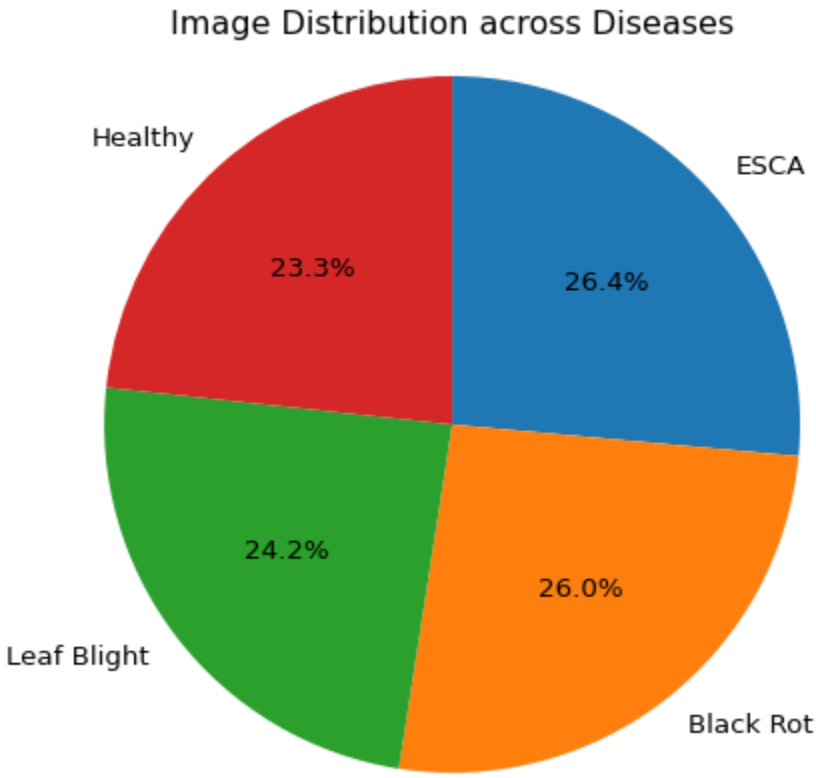

Figure 2: A pie chart depicting the class distribution for the dataset.

Next, we performed data preprocessing. Resizing the images to 224 pixels by 224 pixels allows us to utilize a variety of transfer learning models in the future. We can also augment our data to make our models more accurate on low-quality images and those with unusual backgrounds. Some of the augmentations used include zooming, rotating, illuminating, dimming, and shearing. The final dataset used to train our model contained 1,656 original images and 1,344 augmented images for a total of 3,000 images per class. Augmented images are used to help the model generalize, but they were not included in the testing data so the model could be evaluated on real, high-quality images. This dataset has also been uploaded to Kaggle for the community to access.

3. Methodology

In the proposed framework, grapevine leaf images are used as input. They are normalized, resized, and augmented during data preprocessing procedures. Afterward, a model classifies the

plant as having black rot, Esca, leaf blight, or being healthy. The proposed work involves seven major steps:

A. Data Collection

To find a suitable dataset, we navigated to kaggle.com/datasets to look for a dataset with at least three different classes, a healthy class, and at least 500 images per class. Once the dataset was found, analysis was done on the images in the dataset. It appeared that leaves were taken off grapevines and photographed with a plain background. Afterward, a Kaggle notebook was created, and the dataset was added to the input directory.

B. Exploratory Data Analysis

To get a better feel for the data, we read all the image files and stored them in a dataframe. Next, we plotted a bar graph and pie chart showing the image distribution across classes. In addition, eight images for each class were plotted on a grid. After that, we divided all images' pixel values by 255 so the model trains faster.

C. Data Augmentation

We decided to perform data augmentation to ensure that the final model could handle images with natural backgrounds. The augmentations below were done randomly on the entire dataset:

- Zoom in or out of the image.
- Flip the image horizontally or vertically.
- Rotate the image.
- Increase or decrease the brightness of the image.
- Shear the image.

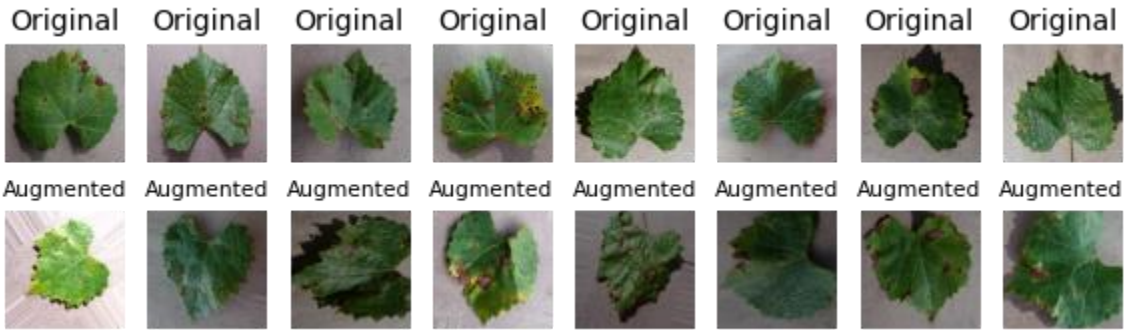

Figure 3: Images depicting the augmentation of grapevine leaves infected with black rot.

Figure 4: Images depicting the augmentation of grapevine leaves infected with Esca.

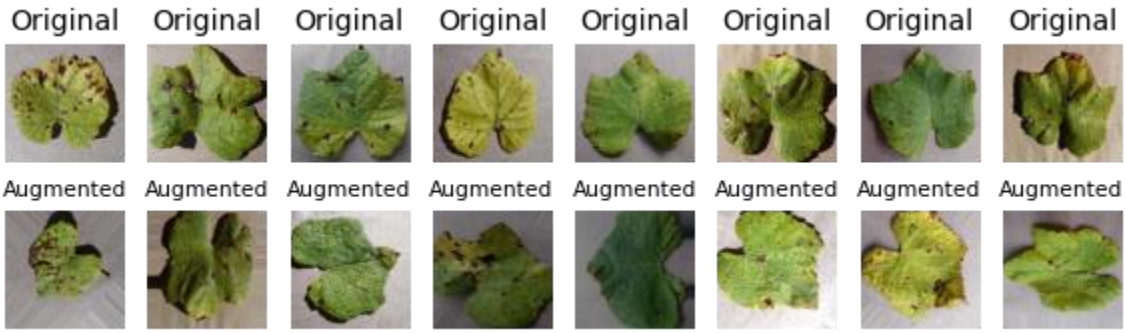

Figure 5: Images depicting the augmentation of grapevine leaves infected with leaf blight.

Figure 6: Images depicting the augmentation of healthy grapevine leaves.

Finally, we combined 1,344 images from the original dataset with 1,656 augmented images for each class. This led to each class having 3,000 images, with 12,000 images in total. This newly created dataset was published on Kaggle.com for others to use.

D. Baseline Model

First, we created the model architecture described below and built it as a "sequential model" in TensorFlow. Next, two fully connected Dense layers with the relu activation function and an output layer with four output neurons were added at the top of the model. At this stage, we checked for overfitting and underfitting using train and validation loss. Additionally, the model's performance was evaluated using validation accuracy, a confusion matrix, and additional metrics. The images that the model predicted inaccurately were visualized to understand the model's pitfalls better.

```
Input(shape=(224,224,3))
Conv2D(32, 6, padding='same', activation='relu')
BatchNormalization()
MaxPooling2D()
Conv2D(32, 5, padding='same', activation='relu')
BatchNormalization()
MaxPooling2D()
Conv2D(32, 4, padding='same', activation='relu')
BatchNormalization()
MaxPooling2D()
Conv2D(32, 3, padding='same', activation='relu')
BatchNormalization()
MaxPooling2D()
Conv2D(32, 3, padding='same', activation='relu')
BatchNormalization()
MaxPooling2D()
Conv2D(32, 3, padding='same', activation='relu')
BatchNormalization()
Conv2D(32, 3, padding='same', activation='relu')
BatchNormalization()
MaxPooling2D()
Dropout(0.2)
Flatten()
Dense(512, activation='relu')
Dense(512, activation='relu')
Dense(4)
```

Figure 7: A description of the baseline CNN's architecture.

E. Transfer Learning

The five models tested using transfer learning are DenseNet121, EfficientNetB7, MobileNetV2, ResNet50, and VGG16.

DenseNet is a convolutional neural network with each layer connected to all other layers in a feed-forward fashion. Its main benefits include mitigating the vanishing gradient problem, increasing feature propagation, and lowering the number of parameters. The three model architectures are DenseNet121, DenseNet169, and DenseNet201. DenseNet121 is the smallest of the three models; it has 121 layers. Its architecture is depicted in the diagram below:

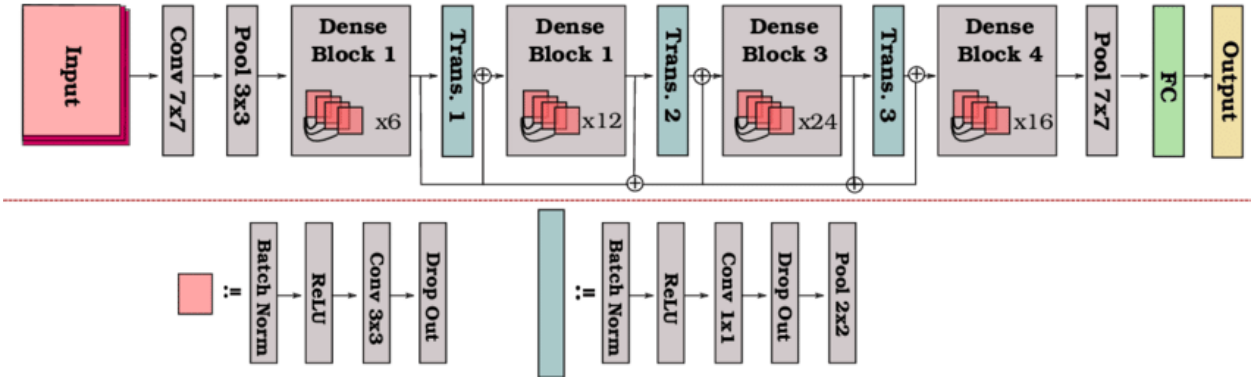

Figure 8: A diagram depicting the architecture of the DenseNet121 model.

EfficientNet is a convolutional neural network that uniformly scales all dimensions (depth, width, resolution) using a compound coefficient. Its main benefits include being able to train efficiently, easily adapt to a wide variety of datasets, and massively reduce the number of parameters in a model. The eight model architectures are EfficientNetB0, EfficientNetB1, EfficientNetB2, EfficientNetB3, EfficientNetB4, EfficientNetB5, EfficientNetB6, and EfficientNetB7. EfficientNetB7 is the largest of the eight models; it has 813 layers. Its architecture is depicted in the diagram below:

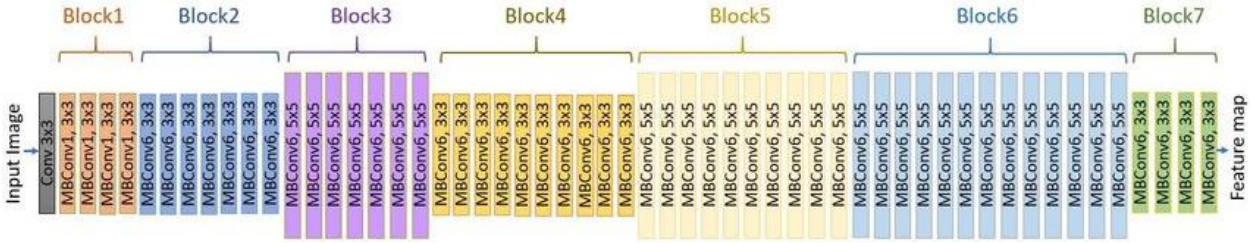

Figure 9: A diagram depicting the architecture of the EfficientNetB7 model.

MobileNet is a convolutional neural network based on an inverted residual structure; the residual block's input and output are thin bottleneck layers. Its main benefits include having a small model size (only about 9MB), requiring less computation due to Depthwise Separable Convolutions, and being compatible with mobile devices since it does not require Graphics Processing Units (GPUs). The three model architectures are MobileNet, MobileNetV2, and MobileNetV3. MobileNetV2 was chosen for this paper; it has 53 layers. Its architecture is depicted in the diagram below:

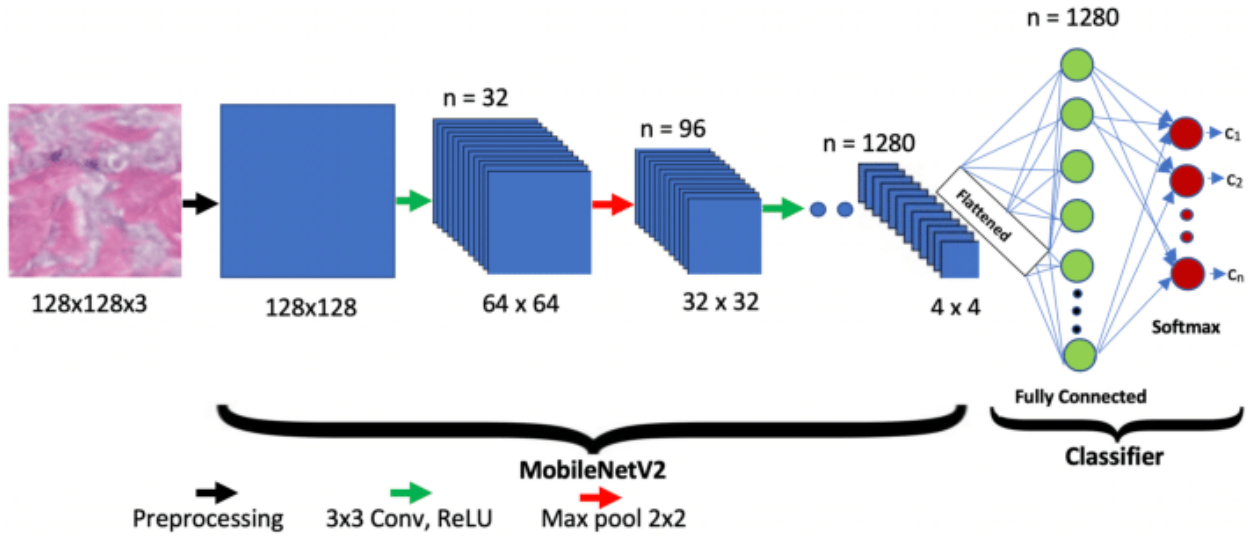

Figure 10: A diagram depicting the architecture of the MobileNetV2 model.

ResNet is a convolutional neural network with a Residual Block. Its main benefits include allowing deep networks to train with high accuracy, minimizing the effect of layers that negatively impact model performance, and resolving the vanishing gradient problem. The three model architectures are ResNet50, ResNet101, and ResNet152. ResNet50 is the smallest of the three models; it has 50 layers. Its architecture is depicted in the diagram below:

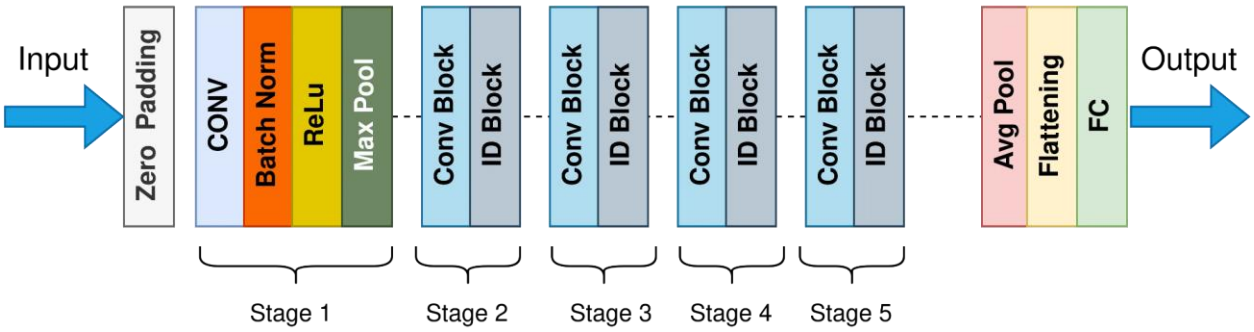

Figure 11: A diagram depicting the architecture of the ResNet50 model.

VGG is a convolutional neural network with few layers but many parameters. Its main benefit is being relatively simple to understand and explain. The two model architectures are VGG16 and VGG19. VGG16 is the smaller of the two models; it has 16 layers. Its architecture is depicted in the diagram below:

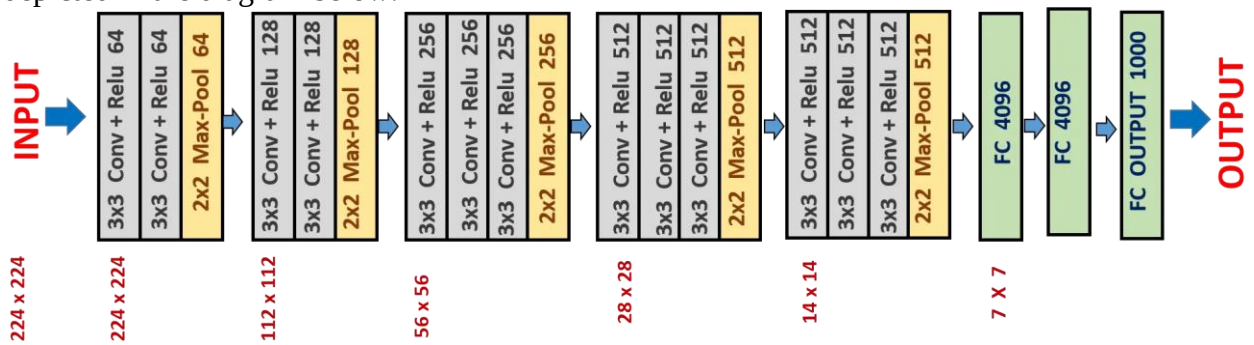

Figure 12: A diagram depicting the architecture of the VGG16 model.

For each transfer learning model, we imported its pre-trained weights from ImageNet. Next, a sequential model containing the transfer learning model was created. A GlobalAvgPool2D layer was then added to the sequential model. By calculating the mean of the input's width and height, this layer performed downsampling. It reduces the total number of parameters and the chance of overfitting. Finally, we added a Dense layer with a softmax activation. By doing this, raw neural network outputs were converted into probability vectors. Next, we set an EarlyStopping callback that stopped training if validation accuracy did not improve after eight epochs. In addition, we implemented a ModelCheckpoint callback that saves the model with the lowest validation loss. The sequential model was compiled with the Adam optimizer. Finally, each model was trained for 25 epochs.

F. Model Evaluation

The most suitable CNN models for the max-voting ensemble were determined using validation accuracies, confusion matrices, and error visualizations. In a max-voting ensemble, each base model makes a prediction on an image. Each prediction counts as a vote, and the disease with the most votes is the final prediction. If there is a tie, the final prediction is decided by the highest scoring base model. The ensemble was evaluated using accuracy score, confusion matrix, classification metrics, and error visualization. We also found out how long the ensemble took to classify a single image with and without a GPU. After that, we downloaded the HDF5 file containing the most accurate models' weights.

G. Website Deployment

In order to develop a website, we started by creating a Google Colab notebook. Next, we logged into an NGrok account and copied the authentication token. It was pasted into the notebook environment to connect to NGrok. Basic code was written that loaded the model and classified an input image as having a certain disease. It was saved as a Python file and run using Streamlit. The link printed in the cell output was used to access the website.

4. Results and Discussion

The three most accurate models had over 99% accuracy. The EfficientNetB7, ResNet50, and baseline CNN models had accuracies of 99.6%, 99.2%, and 99.1% respectively. They were significantly better than the other three models, so the EfficientNetB7, ResNet50, and baseline CNN models were used for the max-voting ensemble. VGG16, DenseNet121, and MobileNetV2 had accuracies of 98.3%, 96.7%, and 94.3% respectively. This information is visualized in the bar graph below.

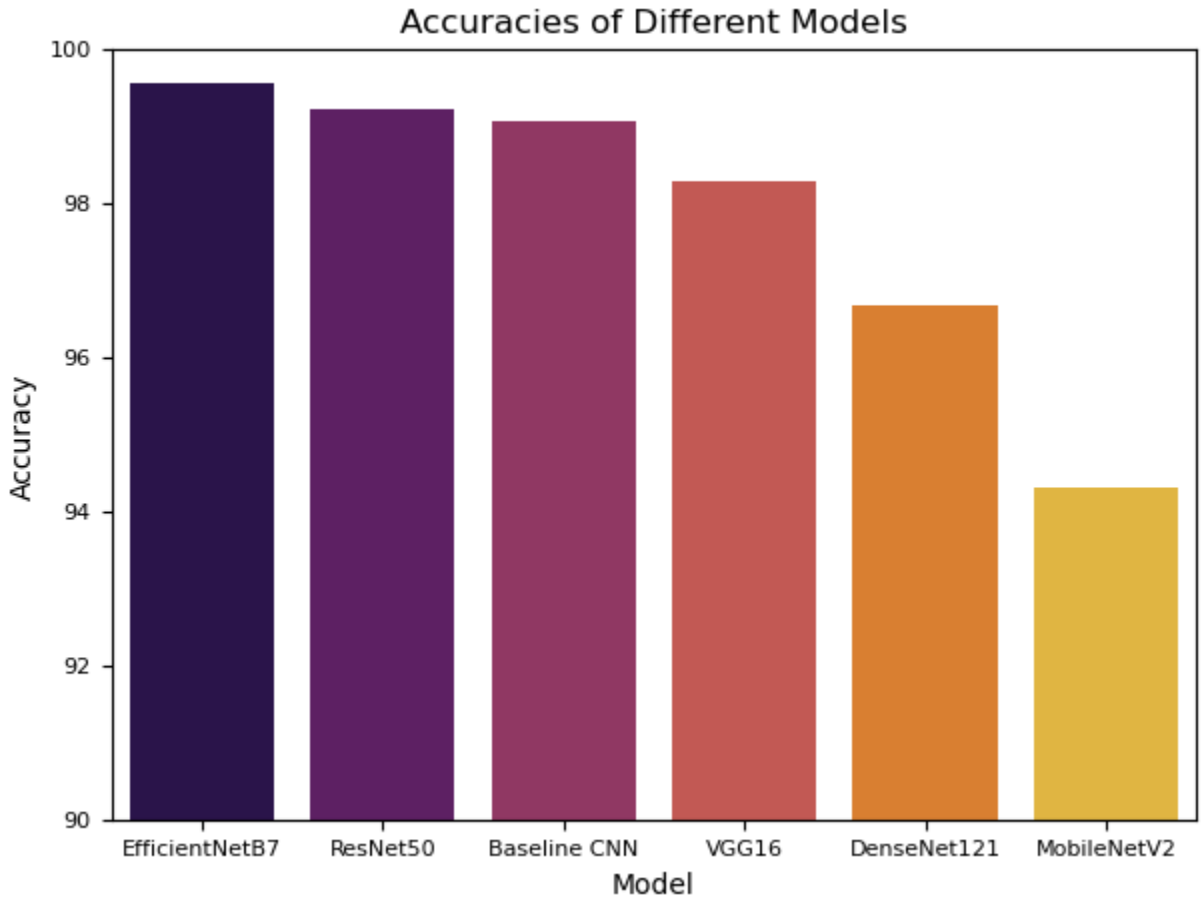

Figure 13: A bar graph depicting the accuracies on testing data of the models tested.

Confusion matrices are tables that provide information regarding model errors. It is especially helpful when there is a drastic class imbalance. However, it always gives insights into a model's strengths and weaknesses. The confusion matrix for each model is shown below.

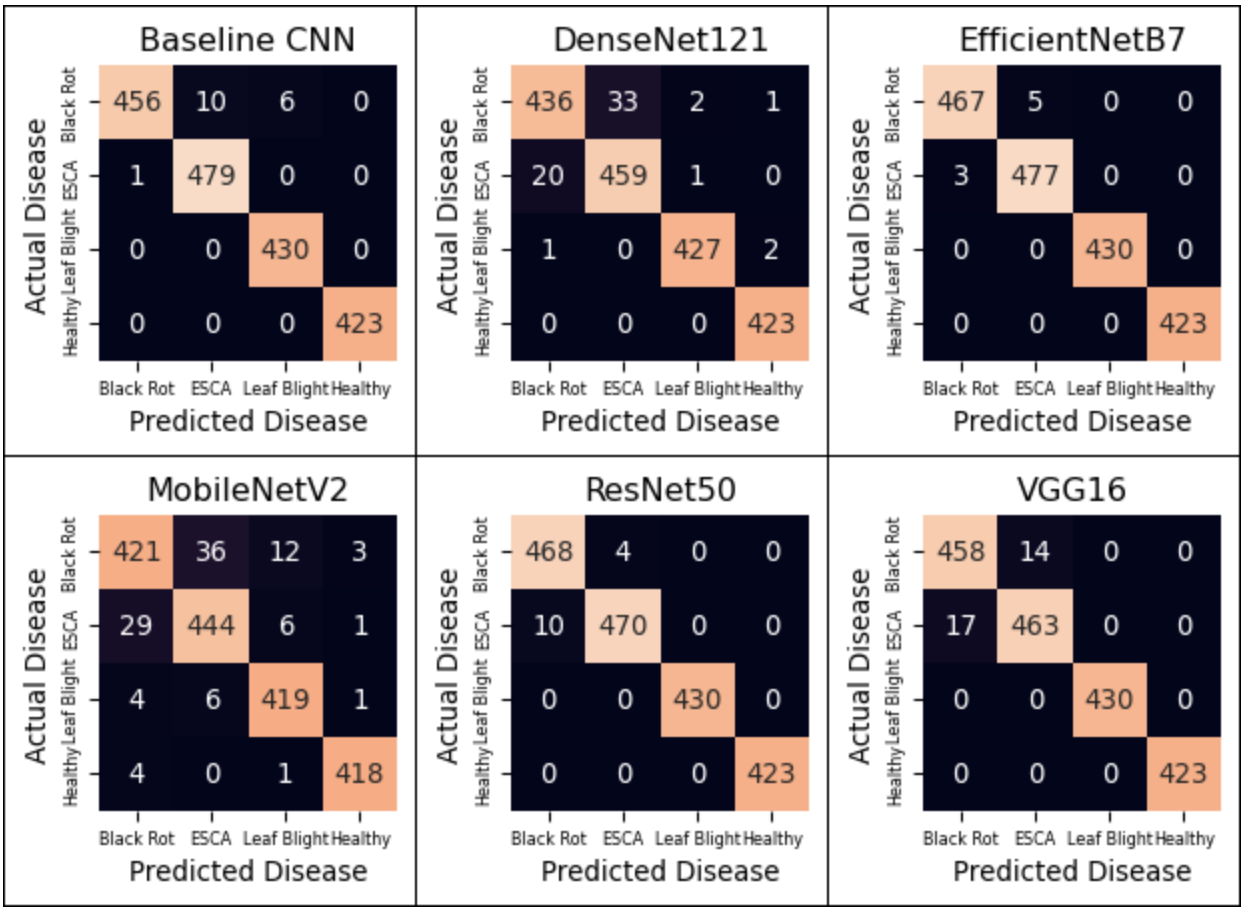

Figure 14: Six confusion matrices depicting errors of the models tested.

From these matrices, it is visually clear that black rot and Esca pictures were easily confused with each other. In particular, DenseNet121, EfficientNetB7, ResNet50, and VGG16 made almost all of their misclassifications with black rot and Esca. We can look at the EfficientNetB7 model's error visualization to understand what is creating the inaccurate classifications in these models.

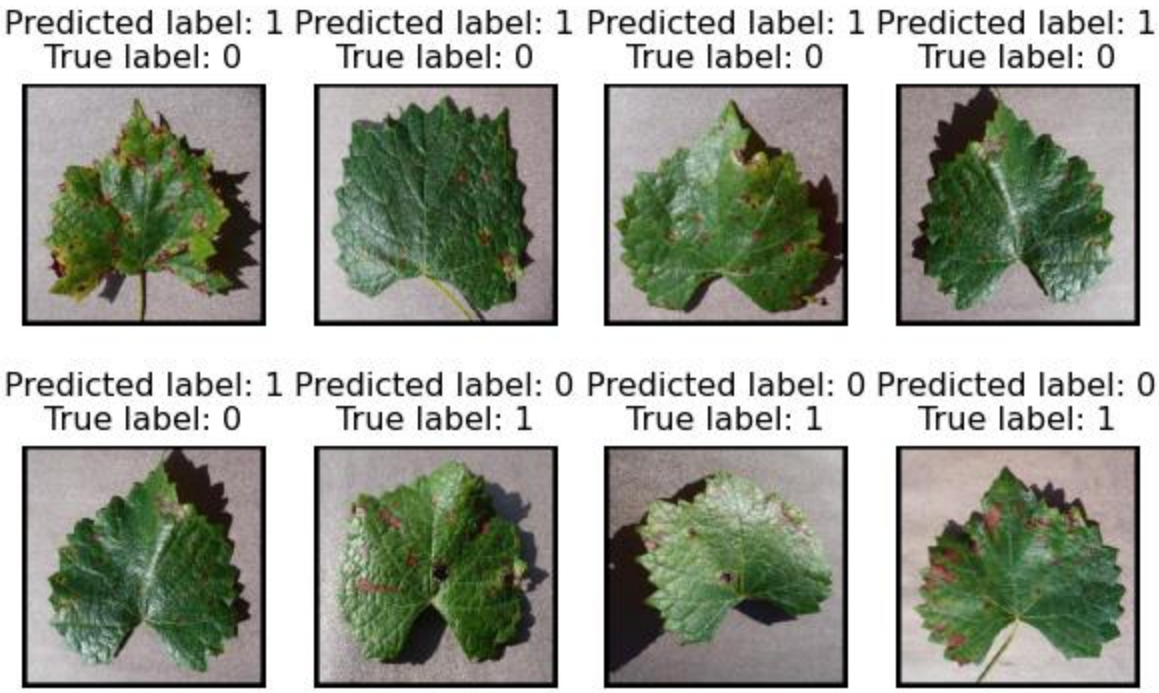

Figure 15: The error visualization of the EfficientNetB7 model. The numbers 0, 1, 2, and 3 refer to black rot, Esca, leaf blight, and healthy respectively.

These images are impossible to classify into two classes. They look almost identical and have the same spot size and shape. It is no surprise that transfer learning models cannot classify them accurately; even a trained visual expert will likely be unable to categorize these images accurately.

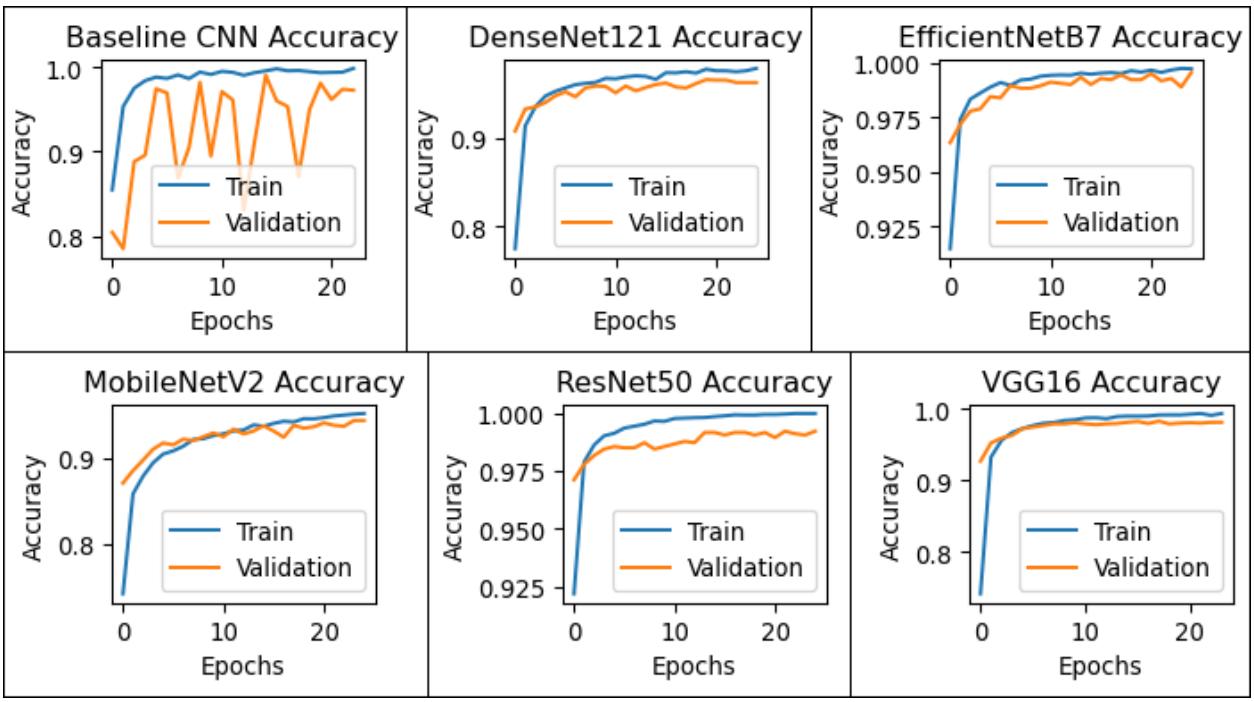

Figure 16: Six line graphs depicting the training and validation accuracies of all models tested.

Next, we gathered insight from training and validation accuracy curves. First, we noticed that the training accuracy is higher than the validation accuracy. This made sense intuitively. However, we noticed that the baseline CNN's validation accuracy during training is inconsistent. This hinted that the model may have overfitted the training data. The rest of the line graphs had no issues.

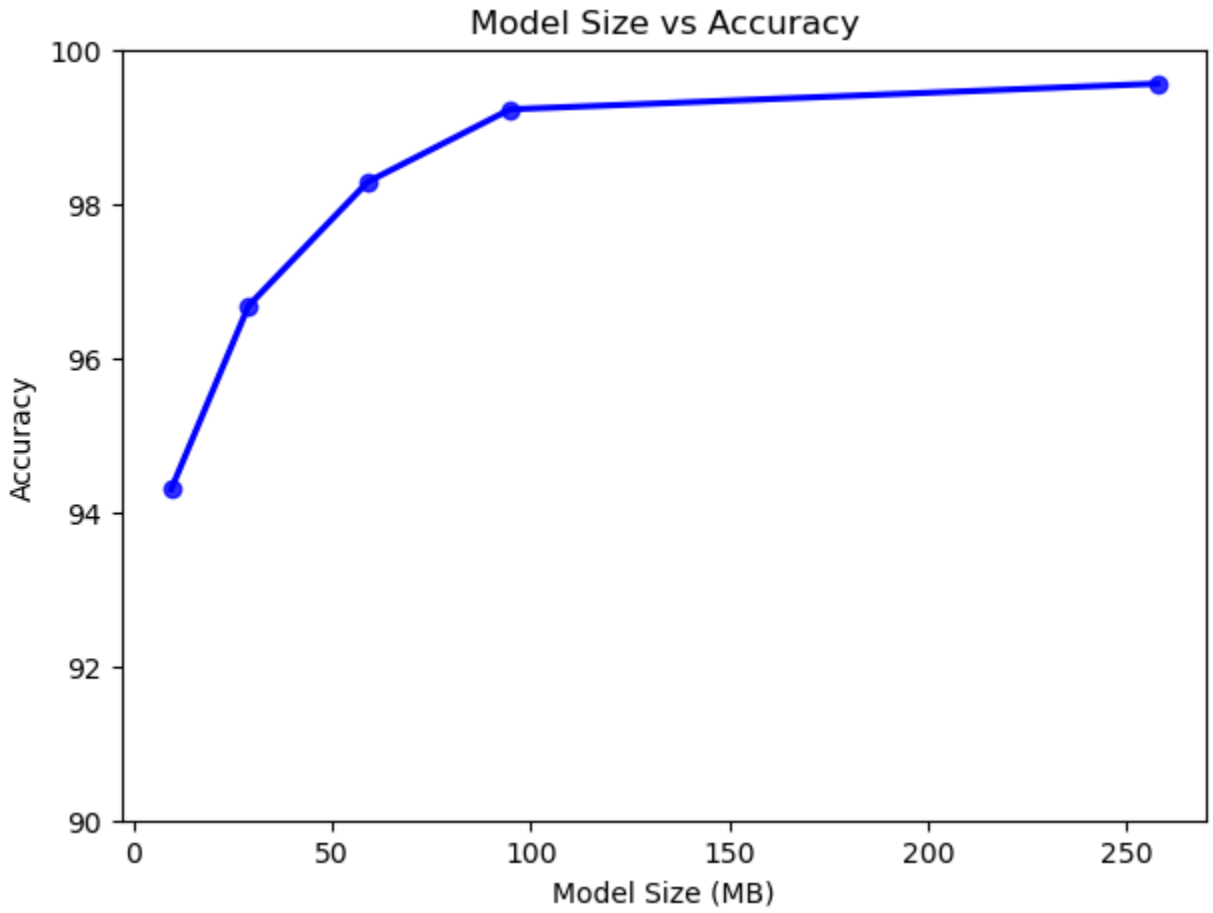

Figure 17: A line graph depicting the correlation between the size of the model's weights in megabytes and the accuracy on testing data.

We also found another noteworthy result. After carefully inspecting the graph, we found that there is a strong correlation between the model weight's file size and accuracy. After about 150 megabytes, even a drastic increase in model size will only yield a small improvement.

The three highest performing models were used in a max-voting ensemble. Its classification metrics, confusion matrix, and error visualization are below.

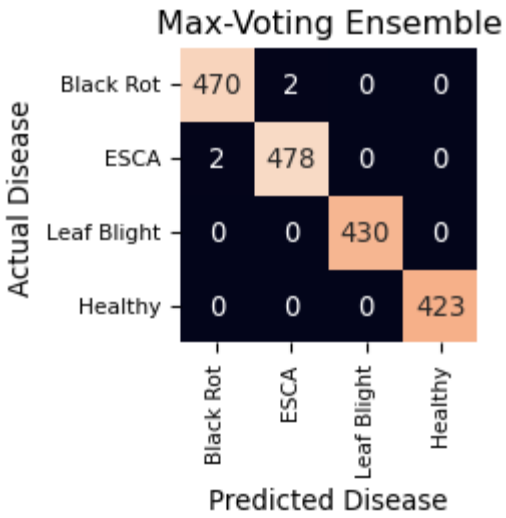

Figure 18: A confusion matrix depicting the errors made by the max-voting ensemble.

	precision	recall	f1-score	support
0	1.00	1.00	1.00	472
1	1.00	1.00	1.00	480
2	1.00	1.00	1.00	430
3	1.00	1.00	1.00	423
accuracy			1.00	1805
macro avg	1.00	1.00	1.00	1805
weighted avg	1.00	1.00	1.00	1805

Figure 19: A table depicting classification metrics for the max-voting ensemble.

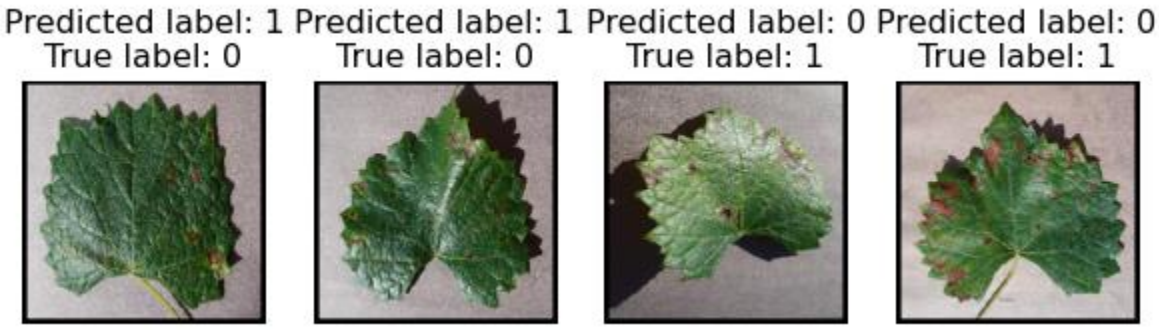

Figure 20: The error visualization for the max-voting ensemble. The numbers 0, 1, 2, and 3 refer to black rot, Esca, leaf blight, and healthy respectively.

The max-voting ensemble performed better on the testing data than all other models; it had an accuracy of 99.8%. The ensemble only had four errors on the testing data. Four images were misclassified as black rot or Esca. To determine its efficiency, the ensemble was timed while classifying images. For a single image, it took the max-voting ensemble an average of 12.17 milliseconds with a GPU and 383 milliseconds without one.

5. Conclusion

In this paper, we learned that EfficientNetB7 and ResNet50 work well for detecting diseases in grapevine plants. Ensembling methods such as max-voting enable us to achieve state-of-the-art accuracy on testing data. A simple model architecture can yield high accuracy, but they are inconsistent across just a few epochs. Finally, we learned that data augmentation is important, as it makes the final model more robust. In addition, we have concluded that models with larger file sizes tend to classify plant diseases more accurately.

Currently, it takes the max-voting ensemble 12.17 milliseconds to predict a single image with a GPU. However, most farmers lack access to GPUs. Without one, it takes the max-voting ensemble 383 milliseconds to predict a single image. This time must be reduced to apply this technology effectively in rural areas with poor farmers. The model requires 224 by 224 pixel images or larger ones for accurate predictions. Having another model that handles extremely low-resolution images may be helpful in rural areas with old-fashioned cameras.

Finally, the users can only input one image at a time on the website. In the future, it is imperative that the website takes thousands of images as input simultaneously for it to be operating at the industrial level. Added functionality would solve this problem. Additionally, a mobile app would be immensely helpful because it could run locally on a farmer's smartphone.

Acknowledgments

We would like to thank Mirna Kheir Gouda for guiding us through the project as a mentor. We are also grateful for Inspirit AI in their offering of the Independent Project Mentorship program, which allowed us to work with Mirna Kheir Gouda.

References

[5] Aloysius, N., & Geetha, M. (2017). *A review on deep convolutional neural networks*. https://doi.org/10.1109/iccsp.2017.8286426

[4] Ansari, A. S., Mustafa, M., Ritonga, M., Jamwal, P., Mohammadi, M. S., Veluri, R. K., Kumar, V., & Shah, M. A. (2022). Improved Support Vector Machine and Image Processing Enabled Methodology for Detection and Classification of Grape Leaf Disease. *Journal of Food Quality, 2022,* 1–6. https://doi.org/10.1155/2022/9502475

[2] Kenfaoui, J., Radouane, N., Mennani, M., Tahiri, A., Ghadraoui, L. E., Belabess, Z., Fontaine, F., Hamss, H. E., Amiri, S., Lahlali, R., & Barka, E. A. (2022). A Panoramic View on

Grapevine Trunk Diseases Threats: Case of Eutypa Dieback, Botryosphaeria Dieback, and Esca Disease. *Journal of Fungi*, *8*(6), 595. https://doi.org/10.3390/jof8060595

[1] Molitor, D., & Beyer, M. (2014). Epidemiology, identification and disease management of grape black rot and potentially useful metabolites of black rot pathogens for industrial applications - a review. *Annals of Applied Biology*, *165*(3), 305–317. https://doi.org/10.1111/aab.12155

[6] Shorten, C., & Khoshgoftaar, T. M. (2019). A survey on Image Data Augmentation for Deep Learning. *Journal of Big Data*, *6*(1). https://doi.org/10.1186/s40537-019-0197-0

[3] Zherdev, A. V., Vinogradova, S., Byzova, N. A., Porotikova, E. V., Kamionskaya, A. M., & Dzantiev, B. B. (2018). Methods for the Diagnosis of Grapevine Viral Infections: A Review. *Agriculture*, *8*(12), 195. https://doi.org/10.3390/agriculture8120195