JOURNAL OF
EMERGING INVESTIGATORS

Article

Vineyard vigilance: Harnessing deep learning for grapevine disease detection

Rajarshi Mandal[1], **Mirna Kheir Gouda** [2]
[1] Lexington High School, Lexington, Massachusetts
[2] Department of Biological Engineering, MIT, Cambridge, Massachusetts

SUMMARY

According to the International Organization of Vine and Wine, an impressive 77.8 million tons of grapes are cultivated globally each year. Grapes, long a cornerstone of both diets and agriculture worldwide, are increasingly threatened by diseases such as black rot, Esca, and leaf blight. In modern grapevine orchards, the prevalent method of disease detection relies on human observation, a process that often leads to delays in identifying these afflictions. This inefficiency typically results in diminished crop yields and inferior 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. In this paper, we deployed Convolutional Neural Networks (CNN), a type of deep learning model, to accurately classify healthy and disease-affected grapevine leaf images. We developed a baseline CNN model along with several transfer learning CNN models, namely DenseNet121, EfficientNetB7, MobileNetV2, ResNet50, and VGG16. We hypothesized that ResNet50 would be the best-performing model as it allows deep networks to train with high accuracy, but EfficientNetB7 turned out to be the most accurate model after experimentation. We created a max-voting ensemble with the three most accurate CNN models from the set, and it delivered better results than any individual CNN model. Our max-voting ensemble was then deployed on the web. 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.

INTRODUCTION

According to the International Organization of Vine and Wine, 77.8 million tons of grapes are grown annually (1). Grapes have been a staple of people's diets since ancient times and are usually used for direct consumption or for making wine or 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 fungicides (2, 3). Black rot is a fungal disease triggered by *Guignardia bidwellii* in hot and humid weather (4). It renders grapes inedible and has been called the "Achilles Heel" of grape production in the Middle East (5). Esca is a trunk disease sometimes caused by the fungus *Phaeoacremonium aleophilum* that usually manifests during July and August (6). In France, 13% of vineyards are affected yearly by Esca, netting a cumulative loss of over 1 billion euros (7). Leaf blight is a bacterial disease caused by *Xylophilus ampelinus* (8). Infected vineyards have reported losses of over 70% of typical yields (9). These diseases are problematic due to increased fungicide usage and decreased crop yields, significantly decreasing grapevine orchard profitability.

These common vineyard diseases are traditionally managed through a combination of cultural practices and fungicides, with black rot being controlled by improving vineyard aeration and sanitation (10), Esca requiring preventive measures due to the absence of effective chemical treatments (11), and leaf blight managed through fungicidal sprays and moisture control (12). Accurate and early detection of vineyard diseases is critical to reduce fungicide usage and minimize crop loss. Traditionally, the detection and diagnosis of these diseases have relied heavily on the expertise of trained professionals who visually inspect the plants (13). However, this method is constrained by the availability of experts and subjective interpretation. This challenge can be innovatively addressed through the application of machine learning, specifically deep learning and Convolutional Neural Networks (CNNs).

Machine learning, a subset of artificial intelligence, enables systems to autonomously learn and improve from experience. Deep learning, a specialized subset of machine learning, employs a neural network, which is a system of interconnected nodes that mimic the neural connections in the human brain, to analyze and learn from data in complex ways (14). This architecture enables the network to recognize patterns and make decisions based on input data, enhancing its ability to understand and process information at a deep level. CNNs, a type of deep neural network, are exceptionally effective in analyzing visual imagery (15).

In the context of viticulture, researchers have explored the use of a machine learning technique named Support Vector Machines (SVMs) for detecting grapevine diseases (16). SVMs are particularly adept at binary classification problems where elements of a dataset are categorized into two distinct groups (17). Consequently, this research was limited to differentiating between healthy and diseased leaves without identifying specific diseases. Moreover, the study used only 250 diseased and 400 total leaf images while delivering slightly more than 95% accuracy, rendering it unsuitable to meet the diverse and practical needs of farmers in managing a range of vineyard diseases.

In contrast to SVMs, CNNs can be trained on extensive datasets of leaf images, enabling them to identify different disease symptoms with higher accuracy. This approach is part of the broader field of precision agriculture, where

https://doi.org/10.59720/23-251

	Training Images (Original)	Training Images (Used)			Testing Images	Total Images
		From Original	Augmented	Total Training		
Black rot	1888	1656	1344	3000	472	3472
Esca	1920	1656	1344	3000	480	3480
Leaf blight	1722	1656	1344	3000	430	3430
Healthy	1692	1656	1344	3000	423	3423
Total	**7222**	**6624**	**5376**	**12000**	**1805**	**13805**

Table 1: Number of images for different diseases in the dataset. The original dataset had a different number of images for each class in the training dataset. In this research, the same number of training images were used for each class from the original data set for training consistency. The same number of augmented images were created for each class for the same reason.

technology is utilized to enhance crop health and productivity (18). Moreover, transfer learning, where a pre-trained neural network is adapted for a new task, has been successfully employed in agriculture. Popular transfer learning models like DenseNet, EfficientNet, MobileNet, ResNet and VGG, originally developed for general image recognition tasks, are being repurposed for agricultural applications (19–23). In addition to these models, the concept of max-voting ensemble is gaining traction. Max-voting involves combining multiple models and using the majority vote of their predictions to make the final decision (24). Max-voting ensemble improves the overall accuracy by leveraging the strengths of diverse models, making it particularly effective in complex tasks like disease detection in agriculture (24).

DenseNet is a CNN with each layer connected to all other layers in a feed-forward fashion (19). This architecture not only mitigates the vanishing gradient problem – where gradients become too small to effectively train the network during backpropagation – but also enhances feature propagation and reduces the overall number of parameters required (19). Among the DenseNet architectures, we chose DenseNet121 due to its balance of depth and complexity.

EfficientNet is a CNN that uniformly scales all dimensions (depth, width, resolution) using a compound coefficient (20). 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 (20). Of the eight available model architectures, we selected EfficientNetB7 due to its enhanced capability to scale depth, width, and resolution uniformly.

MobileNet is a CNN based on an inverted residual structure; the residual block's input and output are thin bottleneck layers (21). 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) (21). Of the three available model architectures, we chose MobileNetV2 for this paper because it struck a balance between efficiency and accuracy.

ResNet is a CNN with a Residual Block (22). 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 (22). Among the ResNet variants, we selected ResNet50 for its relatively lower complexity, which translated to lower computational demand.

VGG is a CNN with few layers but many parameters (23). Its main benefit is being relatively simple to understand and explain (23). The two model architectures are VGG16 and VGG19. We chose VGG16, which is simpler than VGG19 with its 16 layers, for its ease of understanding and implementation.

In this paper, we used a grapevine leaf image dataset from kaggle.com. It contained four classes (black rot, Esca, leaf blight, and healthy) with about 2,000 training images and about 500 testing images in each class (**Table 1**). We augmented the dataset using techniques like zooming, rotating, illuminating, dimming, and shearing (25). We used the augmented dataset with 12,000 total images to make our models more accurate – even for low-quality images and those with unusual backgrounds – than previously published models. We used augmented images to help the model generalize, but they were not included in the testing data so the model could be evaluated on real images provided in the original dataset.

We developed a baseline CNN model along with the transfer learning CNN models noted above, namely DenseNet121, EfficientNetB7, MobileNetV2, ResNet50 and VGG16. We selected a particular architecture for a given transfer learning model with the goal of optimizing accuracy in grapevine leaf disease detection. We hypothesized that ResNet50 would be the best-performing model as it allowed deep networks to train with high accuracy, but EfficientNetB7 turned out to be the most accurate model after experimentation. We then created a max-voting ensemble with the three most accurate CNN models – EfficientNetB7, ResNet50 and baseline CNN – from the set, and it delivered better results than any individual CNN model. We then deployed this max-voting ensemble on the website.

RESULTS

The grapevine dataset had images of leaves infected with black rot, Esca and leaf blight as well as images of healthy leaves (**Figure 1**). We applied data augmentation techniques to increase the dataset size, simulate real-world variability, improve model generalization and reduce overfitting.

Comparing the accuracies of different CNN model architectures was the first test we performed. In the experiments, the three most accurate models each had over 99% accuracy. The EfficientNetB7, ResNet50, and baseline CNN models had accuracies of 99.6%, 99.2%, and 99.1% respectively (**Figure 2A**). They were better than the other three models, so we used the EfficientNetB7, ResNet50, and baseline CNN models for the max-voting ensemble. VGG16, DenseNet121, and MobileNetV2 had accuracies of 98.3%, 96.7%, and 94.3% respectively (**Figure 2A**).

We compared the predictions of the CNN models with the target values to generate the confusion matrices. From the confusion matrices of the six models (**Figure 2B**), 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 looked at the EfficientNetB7 model's error visualization to understand what was creating the inaccurate classifications in these models. These images were difficult to classify into two classes properly as they looked almost identical and had the same spot size and shape. Consequently, the transfer learning models could not classify them accurately.

Next, we gathered insight from training and validation accuracy curves (**Figure 3A**). These curves were created by calculating the accuracies after each training epoch. Finally, by plotting the model weights' file sizes against their corresponding validation accuracies, we found that there was a strong correlation between the model weight's file size and accuracy (**Figure 3B**).

We used the three highest-performing models – EfficientNetB7, ResNet50 and baseline CNN – in a max-voting ensemble. This ensemble looked at the predictions of each of the CNNs and chose the most popular one. Its classification metrics (**Figure 4A**), confusion matrix (**Figure 4B**), and error visualization (**Figure 4C**) showed that 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. It misclassified only four images – two black rot images were classified as Esca and two Esca images were classified as black rot.

To accurately assess the efficiency of the ensemble, we recorded the classification times. Notably, the max-voting ensemble operated within the same system, with the only variable being the presence of a GPU. This addition enhanced performance: with a GPU, the max-voting ensemble averaged 12.17 milliseconds to classify a single image, whereas, without the GPU, the time increased to 383 milliseconds. This contrast underlined the GPU's role in accelerating computational tasks within an identical CPU environment.

DISCUSSION

In this study, we learned that EfficientNetB7 and ResNet50 worked well individually for detecting diseases in grapevine plants. We hypothesized that ResNet50 would be the most accurate as it allowed deep networks to train with high accuracy. While ResNet50 performed very well with 99.2% accuracy, EfficientNetB7 was even more accurate with 99.6% accuracy. EfficientNetB7, with its ability to train efficiently, easily adapted to a wide variety of datasets, and massively reduced the number of parameters in a model, outperforming all other models. Moreover, ensemble methods such as max-voting enabled us to achieve 99.8% accuracy on testing data with better results than any single transfer learning CNN model.

From the training and validation accuracy curves (**Figure 3A**), 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 was inconsistent. This hinted that the model might have overfitted the training data. The rest of the line graphs (**Figure 3A**) had no issues.

The validation accuracy curve of the baseline CNN model highlighted that simpler model architectures could achieve high accuracy but might exhibit variability across epochs due to overfitting to the training dataset. In contrast, the five transfer learning CNN models, which were derived from the TensorFlow library (an open-source software library developed by the Google Brain team for machine learning and deep neural network research) and not from the baseline CNN model, demonstrated more consistent validation accuracies. These models were carefully modified to accommodate the grapevine disease detection dataset, ensuring their performance metrics are distinct and independent from those of the baseline CNN model. The robust architectures of these transfer learning models, which had been pre-trained on large and diverse datasets, contributed to their stability and reliability in validation accuracy, underscoring their effectiveness for this application.

In this study, data augmentation improved the performance of lower-accuracy models, such as MobileNetV2, DenseNet121, and VGG16, suggesting these models initially suffered from overfitting and benefited from the more varied training examples. Conversely, the higher-performing models – EfficientNetB7, ResNet50, and the baseline CNN – showed only a slight increase in accuracy with augmented data, reflecting their inherent robustness and effective generalization from the training set. These observations highlighted the role of data augmentation in boosting model performance, especially by mitigating overfitting in models that were less optimized.

The correlation between model size and accuracy (**Figure 3B**) presented a nuanced insight into the effectiveness of different models. The observed trend indicated that larger models generally achieve higher accuracy. We could attribute this correlation to several factors that impact a model's size, including the depth and number of layers, the complexity of the network architecture, and the volume of parameters and weights that the model must learn during training. Larger models like EfficientNetB7 and ResNet50, which had more extensive and intricate architectures, were capable of learning more complex features and patterns in the data, contributing to their higher accuracy rates.

However, the relationship between model size and accuracy was not linear. As indicated in our results, beyond a threshold of around 150 megabytes, the improvements

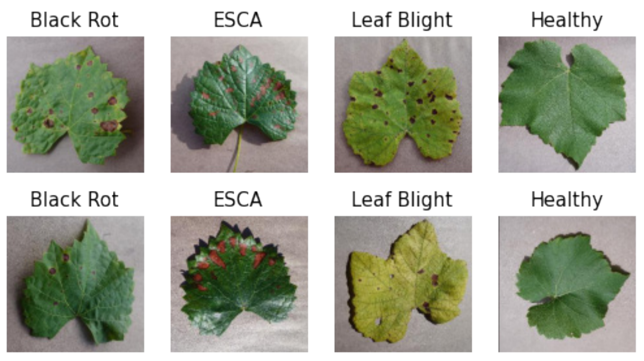


Figure 1: Examples of images showing grapevine leaves that are infected with black rot, Esca, leaf blight, or are healthy. There are 2 images for each class of grapevine leave diseases that are photographed on a plain neutral colored background, one at a time.

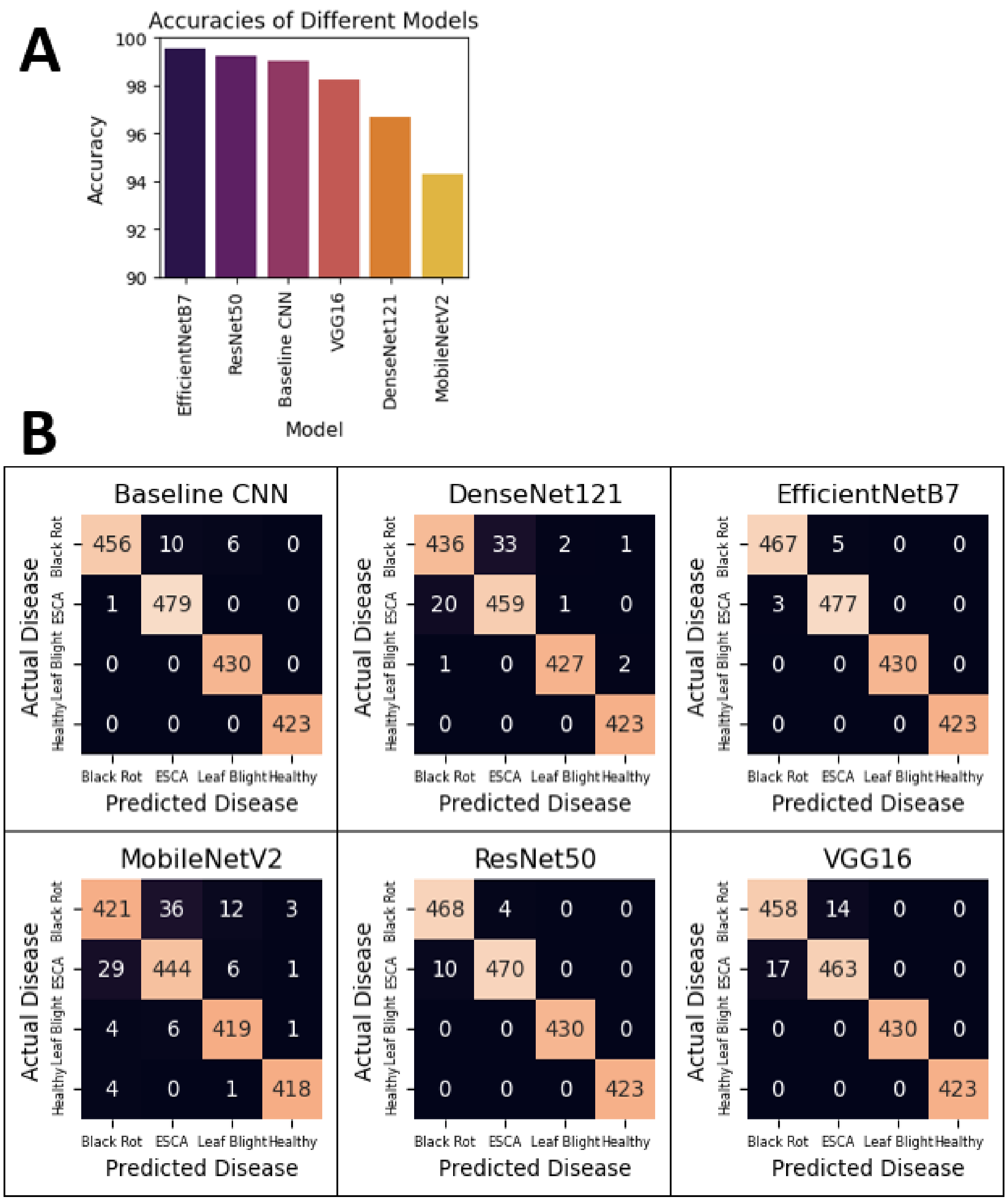
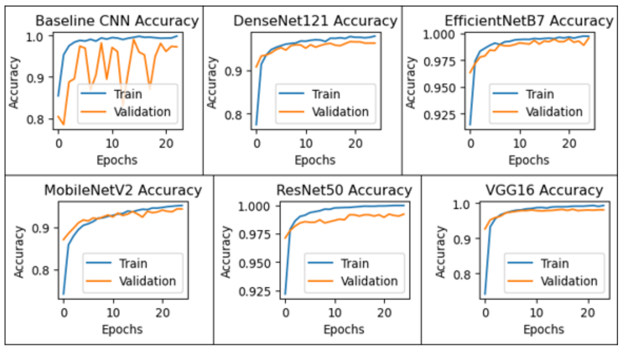


Figure 2: Prediction accuracies and confusion matrices of the models tested. (A) Prediction accuracy on the grapevine leaves testing data. The bar graph shows the accuracies of six different models: Baseline CNN, DenseNet121, EfficientNetB7, MobileNetV2, ResNet50, and VGG16. The accuracies ranged from 94.3% to 99.6%. The models are sorted in decreasing order based on their respective accuracy. (B) Six confusion matrices depicting errors of the models tested. For each confusion matrix, the x-axis has the predicted disease while the y-axis has the actual disease. The number of testing images that fall into a given category is written inside each square. The squares that are colored orange represent a large number of data points and black represent a small number of data points.

in accuracy became marginal (**Figure 3B**), which was particularly revealing. This plateau effect suggested that while more complex models with a greater number of parameters could capture nuanced patterns in data more effectively, there was a point beyond which additional complexity did not yield significant benefits. This could be due to overfitting, where the model became so well-tuned to the training data that it failed to generalize effectively to new, unseen data. It also reflected the principle of diminishing returns in machine learning, where the cost (in terms of computational resources and time) of increasing model complexity did not always translate into proportional improvements in performance.

Our findings highlighted that the relationship between model size and accuracy significantly affected the feasibility of deploying machine learning models in real-world agricultural contexts. Our research goals were to develop models

A


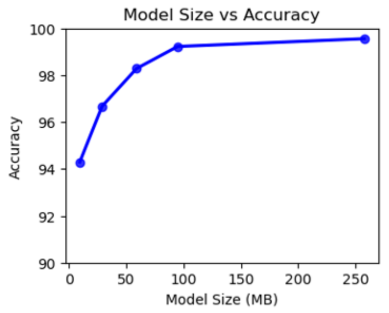
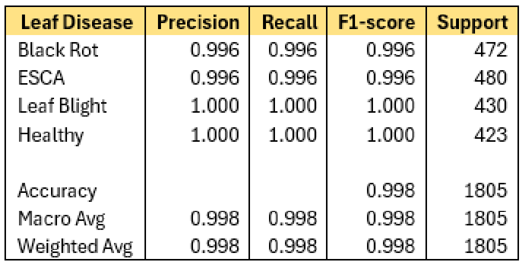
B


Figure 3: Training and validation accuracies of different models as well as the relationship between model size and accuracy. (A) Six line graphs depicting the training and validation accuracies of all models tested. The x-axis represents the epoch number, and the y-axis represents the accuracy. The blue line represents the training accuracy, and the orange line represents the validation accuracy. (B) A line graph depicting the positive correlation between the size of the model's weights in megabytes and the accuracy (in %) on testing data.

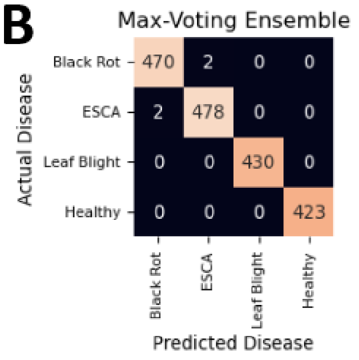
that not only exhibited high accuracy but also maintained computational efficiency for ease of use in the field. This balance ensured that the models were sophisticated enough to detect subtle indicators of disease effectively yet streamlined enough to be deployed on platforms with limited computational power, which was often the case in agricultural environments. The findings from this study would inform the selection of future models, guiding a targeted approach that prioritized both precision and practicality for end-users such as vineyard operators.

In this study, it took 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 took the max-voting ensemble 383 milliseconds to predict a single image. In future, this time could be reduced to apply this technology effectively in rural areas with poor farmers. To enhance model inference time in future work, we plan to implement two key strategies. The first is model pruning, a process where redundant network parameters are identified and removed. This approach aims to streamline the model without sacrificing its original accuracy, by focusing on the most crucial aspects of the network. The second strategy involves the use of depthwise separable convolutions. This technique separates the spatial and depth convolutions, significantly reducing the computational load and thus speeding up the inference process. However, there is some uncertainty regarding how this change might affect the model's ability to accurately capture complex features, and consequently, its overall accuracy.

Additionally, it is important to note that previous studies typically had not reported on classification times. This omission was largely because such timings were heavily influenced

A


Leaf Disease	Precision	Recall	F1-score	Support
Black Rot	0.996	0.996	0.996	472
ESCA	0.996	0.996	0.996	480
Leaf Blight	1.000	1.000	1.000	430
Healthy	1.000	1.000	1.000	423
Accuracy			0.998	1805
Macro Avg	0.998	0.998	0.998	1805
Weighted Avg	0.998	0.998	0.998	1805

B


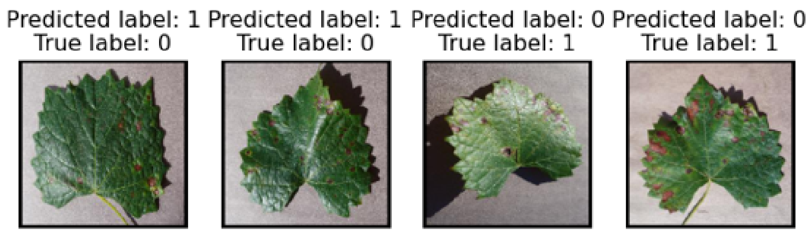
C


Figure 4: Classification metrics, confusion matrix and error visualization for max-voting ensemble deep learning model. (A) Classification metrics for the max-voting ensemble including precision, recall, f1-score, accuracy, macro average, and weighted average. (B) A confusion matrix depicting the errors made by the max-voting ensemble. The x-axis has the predicted disease while the y-axis has the actual disease. The number of testing images that fall into a given category is written inside each square. These squares are colored orange representing a large number of data points and black representing a small number of data points. (C) All of the grapevine leaf images that were misclassified by the max-voting ensemble are depicted. The predicted label and true label are shown above the image. The numbers 0, 1, 2, and 3 refer to black rot, Esca, leaf blight, and healthy respectively. Out of the 4 misclassified images, 2 black rot-affected images were predicted as Esca, and another 2 Esca-affected images were classified as black rot.

https://doi.org/10.59720/23-251

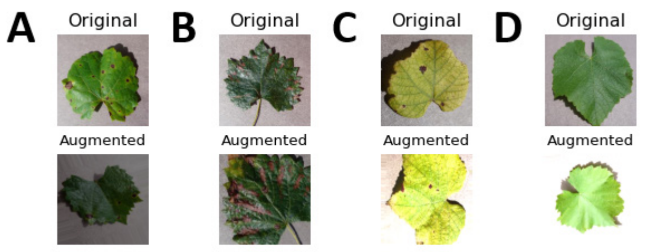


Figure 5: Original and augmented images of different disease-affected and healthy leaves. (A) Black rot-infected grapevine leaves. (B) Esca-infected grapevine leaves. (C) Leaf blight-infected grapevine leaves. (D) Healthy grapevine leaves.

by the specific underlying hardware platform, including the CPUs and GPUs available to a researcher. Consequently, direct comparisons of classification speed across different studies were challenging without a standardized hardware benchmark.

The model required 224 by 224-pixel images or larger ones for accurate predictions. Having another model that handles extremely low-resolution images might 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.

In summary, this research successfully demonstrated the superior performance of EfficientNetB7 and ResNet50 in detecting grapevine diseases, with data augmentation proving particularly beneficial for models susceptible to overfitting. While ensembling methods like max-voting significantly enhanced accuracy, the study also revealed a diminishing returns effect in the relationship between model size and accuracy, guiding practical model deployment in agriculture. Future work aims to reduce inference time through model pruning and depthwise separable convolutions, crucial for application in resource-constrained settings. Enhancing the website for bulk processing and considering a mobile app for local processing present further opportunities to increase the utility and accessibility of this technology in agricultural practices.

MATERIALS AND METHODS

Our study involved seven major steps that included data collection, exploratory data analysis, data preprocessing and augmentation, baseline CNN model development, transfer learning, model evaluation and website deployment.

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. We found a satisfactory dataset, titled "Grape_disease," that was published by the kaggle.com user "Pushpa Lama" on July 1, 2021, at 14:33:41 (Eastern Daylight Time). It contained four classes (black rot, Esca, leaf blight, and healthy) with about 2,000 training images and about 500 testing images in each class (**Table 1**). These original 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, the original dataset had 7,222 training images and 1,805 testing images (**Table 1**). Afterward, we created a kaggle.com notebook and added the dataset to the input directory.

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. We found that black rot, Esca, leaf blight, and healthy images were 26.0%, 26.4%, 24.2%, and 23.3%, respectively, of the total images in the original dataset. In addition, eight images for each class were plotted on a grid. It appeared that leaves were taken off grapevines and photographed with a plain background.

Data Preprocessing and Augmentation

Next, we performed data preprocessing. Resizing the images to 224 pixels by 224 pixels allowed us to utilize a variety of transfer learning models in the subsequent steps. Additionally, we normalized the images by dividing all pixel values by 255, a common practice in image processing. This step scaled the pixel values to a range of 0 to 1, facilitating the model's learning process by ensuring numerical stability and speeding up convergence during training. Normalizing the data in this way helped to achieve faster, more efficient training by reducing the computational burden on the model. We decided to perform data augmentation to ensure that the final model could handle low-resolution images as well as images with natural backgrounds. We did the augmentations (**Figures 5A – D**), randomly on the entire dataset by techniques like zooming in or out of the image, flipping the image horizontally or vertically, rotating the image, increasing or decreasing the brightness of the image, and shearing the image.

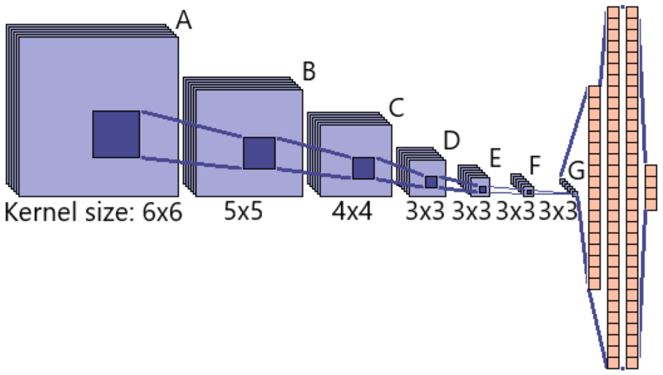


Figure 6: The baseline CNN model architecture. The input layer is denoted with the letter A and has dimensions 224x224x3. The convolutional layers with letters B, C, D, E, F, and G have dimensions 112x112x32, 56x56x32, 28x28x32, 14x14x32, 7x7x32, and 3x3x32 respectively. The kernel size is shown underneath the visual representation of the corresponding convolutional layer. All convolutional layers are shown along with the kernel size. The max-pooling from each convolutional layer to the next causes the width and height to halve. Finally, there are three hidden layers with 288, 512, and 512 nodes respectively and an output layer with 4 nodes. These layers are represented with a column of small orange boxes.

https://doi.org/10.59720/23-251

Finally, we combined 1,656 images from the original dataset with 1,344 augmented images for each class. Please note that we used 1,656 training images for each class as the original dataset had a different number of training images in each class, and we wanted to ensure the models were trained equally for each class. This led to each class having 3,000 images, with 12,000 images in total (**Table 1**). We published this newly created dataset on kaggle.com for others to use. Augmented images were used to help the model generalize, but they were not included in the testing data so the model could be evaluated on the real images present in the dataset.

Baseline CNN Model

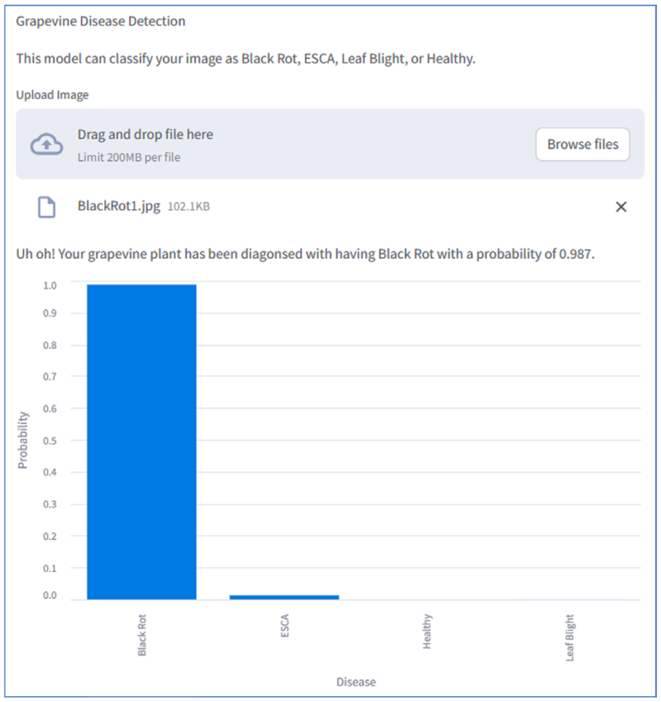
First, we created the model architecture (**Figure 6**) and built it as a "sequential model" in TensorFlow. Next, we added two fully connected dense layers with the relu activation function and an output layer with four output neurons at the top of the model (**Figure 6** and **Appendix**). At this stage, we checked for overfitting and underfitting using training and validation loss. Additionally, we evaluated the model's performance using validation accuracy, a confusion matrix, and a classification report. The images that the model predicted inaccurately were visualized to understand the model's pitfalls better. We used Python Version 3.0.12 for programming, Jupyter Notebook Version 7.0.6 for development, and various Python libraries including TensorFlow, seaborn, scikit-learn, pandas, numpy and matplotlib for CNN modeling (**Appendix**).

Transfer Learning

The five CNN models tested using transfer learning were DenseNet121, EfficientNetB7, MobileNetV2, ResNet50, and VGG16. For each transfer learning model, we imported its pre-trained weights from ImageNet. Next, we created a sequential model containing the transfer learning model. We then added a GlobalAvgPool2D layer to the sequential model. By calculating the mean of the input's width and height, this layer performed downsampling. It reduced 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 saved the model with the lowest validation loss. We compiled the sequential model with the Adam optimizer. Finally, we trained each model for 25 epochs, aligning with established conventions in machine learning. This duration struck a balance between undertraining and overfitting: too few epochs might prevent the model from fully learning from the dataset, while too many could cause the model to learn the training data too well, failing to generalize to new data.

Model Evaluation

We determined the most suitable CNN models for the max-voting ensemble using validation accuracies, confusion matrices, and error visualizations. In a max-voting ensemble, each base model made a prediction on an image. Each prediction counted as a vote, and the disease with the most votes was the final prediction. If there was a tie, the final prediction was decided by the highest-scoring base model. We evaluated the ensemble 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.


Grapevine Disease Detection

This model can classify your image as Black Rot, ESCA, Leaf Blight, or Healthy.

Upload Image

Drag and drop file here
Limit 200MB per file

BlackRot1.jpg 102.1KB

Uh oh! Your grapevine plant has been diagonsed with having Black Rot with a probability of 0.987.

Figure 7: Grapevine disease detection website in action. A grapevine leaf that is affected with black rot has been uploaded. The user learned that the leaf is infected with black rot with 0.987 probability.

Website Deployment

To develop a website, our code loaded the final model and classified an input image as having a certain disease. It was saved as a Python file and run using Streamlit. We used the link printed in the cell output to access the website. We uploaded various grapevine leaf images to the website to test the functionality (**Figure 7**).

Received: September 30, 2023
Accepted: January 26, 2024
Published: August 21, 2024

REFERENCES

1. "2019 Statistical Report on World Vitiviniculture." *International Organisation of Vine and Wine*, 2019, www.oiv.int/public/medias/6782/oiv-2019-statistical-report-on-world-vitiviniculture.pdf. Accessed 23 Mar. 2024.
2. Szabó, Márton, et al. "Black Rot of Grapes (Guignardia bidwellii) – A Comprehensive Overview." *Horticulturae*, vol. 9, no. 2, 2023, https://doi.org/10.3390/horticulturae9020130
3. Beris, Evangelos, et al. "Overview of the Esca Complex as an Increasing Threat in Vineyards Worldwide: Climate Change, Control Approaches and Impact on Grape and Wine Quality." *Recent Advances in Grapes and Wine Production - New Perspectives for Quality Improvement*,

edited by António M. Jordão, Renato Botelho, and Uros Miljic, IntechOpen, 2023, https://doi.org/10.5772/intechopen.105897

4. Molitor, D., and M. Beyer. "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*, vol. 165, no. 3, Sept. 2014, pp. 305–317, https://doi.org/10.1111/aab.12155
5. Wilcox, Wayne F. "Grape Disease Control, 2016." *Department of Plant Pathology, Cornell University, NY State Agricultural Experiment Station, Geneva, NY*, 2016, nygpadmin.cce.cornell.edu/pdf/newsletter_notes/pdf72_pdf.pdf. Accessed 23 Mar. 2024.
6. Kenfaoui, J., et al. "A Panoramic View on Grapevine Trunk Diseases Threats: Case of Eutypa Dieback, Botryosphaeria Dieback, and Esca Disease." *Journal of Fungi*, vol. 8, no. 6, June 2022, p. 595, https://doi.org/10.3390/jof8060595
7. Ouadi, L., et al. "Ecophysiological Impacts of Esca, a Devastating Grapevine Trunk Disease, on Vitis Vinifera L." *PLoS ONE*, vol. 14, no. 9, 2019, e0222586, https://doi.org/10.1371/journal.pone.0222586
8. Komatsu, T., and Kondo, N. "Winter Habitat of Xylophilus Ampelinus, the Cause of Bacterial Blight of Grapevine, in Japan." *Journal of General Plant Pathology*, vol. 81, 2015, pp. 237-242. https://doi.org/10.1007/s10327-015-0581-3
9. FMC Corporation. "Grapevine Trunk Diseases Rob Vineyards of Their Most Productive Years." *FMC Ag US*, 10 Aug. 2020, ag.fmc.com/us/en/agronomic-updates/grapevine-trunk-diseases-rob-vineyards-their-most-productive-years. Accessed 23 Mar. 2024.
10. Rid, M., et al. "Management of Grape Diseases: Towards Environmentally Acceptable Methods." *Journal of Plant Diseases and Protection*, vol. 115, no. 5, 2008, pp. 221-234.
11. Fontaine, F., et al. "Esca of Grapevine and Brown Wood Streaking: Foliar Symptoms, Histological Alterations, and Disease Progression." *Phytopathology*, vol. 100, no. 5, 2010, pp. 487-495.
12. Wilcox, W. F. "Advances in Controlling Vineyard Fungal Diseases." *Viticulture and Enology Science*, vol. 62, no. 2, 2007, pp. 141-147.
13. Zherdev, A., et al. "Methods for the Diagnosis of Grapevine Viral Infections: A Review." *Agriculture*, vol. 8, no. 12, Dec. 2018, p. 195, https://doi.org/10.3390/agriculture8120195.
14. LeCun, Yann, et al. "Deep Learning." *Nature*, vol. 521, no. 7553, 2015, pp. 436-444, https://doi.org/10.1038/nature14539
15. Krizhevsky, Alex, et al. "ImageNet Classification with Deep Convolutional Neural Networks." *Advances in Neural Information Processing Systems*, vol. 25, 2012.
16. Ansari, A., et al. "Improved Support Vector Machine and Image Processing Enabled Methodology for Detection and Classification of Grape Leaf Disease." *Journal of Food Quality*, vol. 2022, July 2022, pp. 1–6, https://doi.org/10.1155/2022/9502475
17. Cortes, Corinna, and Vladimir Vapnik. "Support-Vector Networks." *Machine Learning*, vol. 20, no. 3, 1995, pp. 273–297, image.diku.dk/imagecanon/material/cortes_vapnik95.pdf. Accessed 23 Mar. 2024. https://doi.org/10.1007/BF00994018
18. Jones, H. G. "Precision Agriculture and the Future of Farming in Europe." *Scientific and Technical Research Reports*, European Parliament, 2017.
19. Huang, Gao, et al. "Densely Connected Convolutional Networks." *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017. https://doi.org/10.1109/CVPR.2017.243
20. Tan, Mingxing, and Quoc V. Le. "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks." *arXiv*, Cornell University, 28 May 2019, arxiv.org/abs/1905.11946. Accessed 23 Mar. 2024.
21. Howard, Andrew, et al. "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications." *arXiv*, Cornell University, 17 Apr. 2017, arxiv.org/abs/1704.04861. Accessed 23 Mar. 2024.
22. He, Kaiming, et al. "Deep Residual Learning for Image Recognition." *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016. https://doi.org/10.1109/CVPR.2016.90
23. Simonyan, Karen, and Andrew Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition." arXiv, Cornell University, 4 Sep. 2014, arxiv.org/abs/1409.1556. Accessed 23 Mar. 2024.
24. Sarkar, Dipayan, and Vijayalakshmi Natarajan. "Max-Voting." *Ensemble Machine Learning Cookbook*. O'Reilly Media, oreilly.com/library/view/ensemble-machine-learning/9781789136609/6571afd0-0bac-4bb6-9698-c68504baebae.xhtml. Accessed 23 Mar. 2024.
25. Shorten, C., and T.M. Khoshgoftaar. "A survey on Image Data Augmentation for Deep Learning." *Journal of Big Data*, vol. 6, no. 1, July 2019, https://doi.org/10.1186/s40537-019-0197-0

https://doi.org/10.59720/23-251

APPENDIX

The baseline CNN model architecture in Python code is as follows:

```
<code>
# Building the baseline CNN model
baseline_CNN_model = Sequential([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)])
</code>
```

The Input, Conv2D, BatchNormalization, MaxPooling2D, Dropout, Flatten, and Dense layers are predefined within the TensorFlow library. For the Input layer, we specify the shape of the input pictures. For the Conv2D layers, we specify the number of filters, kernel size, padding, and activation. For the Dropout layer, we specify the relative number of connections that we want to reset. For the Dense layers, we specify the number of neurons along with the activation function. Finally, we used the default parameter values for BatchNormalization, MaxPooling2D, and Flatten layers.

The entire code for this research project, including how to deploy the model to a website, can be found at github.com/rajarshi-mandal/grapevine-disease-detection. The original dataset can be found at kaggle.com/datasets/rm1000/grape-disease-dataset-original. The augmented dataset can be found at kaggle.com/datasets/rm1000/augmented-grape-disease-detection-dataset.