Leveraging UAV-Based Hyperspectral Data and Machine Learning Techniques for the Detection of Powderly Mildew in Vineyards

A. Sherafat[1], M. Acosta[1], C. Gonzalez[2], Subodh Bhandari[3], and Amar Raheja[1]

[1]Department of Computer Science; [2]Department of Plant Science; [3]Department of Aerospace Engineering
California State Polytechnic University, Pomona, CA, USA

A paper from the Proceedings of the
16th International Conference on Precision Agriculture
21-24 July 2024
Manhattan, Kansas, United States

Abstract.

This paper presents the development and validation of machine learning models for the detection of powdery mildew in vineyards. The models are trained and validated using custom datasets obtained from unmanned aerial vehicles (UAVs) equipped with a hyperspectral sensor that can collect images in visible/near-infrared (VNIR) and shortwave infrared (SWIR) wavelengths. The dataset consists of the images of vineyards with marked regions for powdery mildew, annotated using LabelImg. Model training is accomplished using neural networks, XGBoost, and stacking. Different vegetation indices calculated using the hyperspectral data such as normalized difference vegetation index (NDVI) are used for the model training along with the data collected from proximal sensors that include CM 1000 Chlorophyll Meter. Expert visual rating of disease severity is also used for training the models. The models offer mapping functionality to determine the exact position of the detected plants. For the model validation, a different set of remote sensing, proximal sensor, and visual inspection data are used. By integrating the Segment Anything Model (SAM) for precise segmentation and fine-tuning the YOLOv10 model for accurate vine tree detection in drone imagery, segmentation and object detection will improve. Performance of the models trained using the three techniques are compared.

Keywords.
Machine Learning, Hyperspectral Data, Powderly Mildew, Prediction, Plant Health.

Introduction

UAV-based hyperspectral sensing combined with machine learning techniques have potential for monitoring and early detection of powdery mildew (PM) in grapes and can be an integral part of an Integrated Pest Management program for viticulture. UAVs can cover a large area in a short amount of time. They can provide high resolution data for the detection of diseases throughout the crop growth season at a low cost. Despite these potentials, UAV technologies has not yet

seen widespread adoption by farmers for the detection and treatment of diseases. Traditional methods of PM detection involves visual inspection [3]. But, for the agronomist to be able to detect, the PM symptoms must already be visible when the damage is already done and disease is already spreading. Human eyes can only see the visible light of the electromagnetic spectrum from 350 nm to 700 nm. Near infrared (NIR) and shortwave infrared (SWIR) sensors can see the reflected light in 700 nm to 2500 nm spectral range, thereby helping early detection of PM and other diseases [4]. When grape is subjected to PM stress, its spectral reflectance changes according to physiological and biochemical changes in its leaves, such as decreased chlorophyll content or destroyed cell or structure [5] or water stress.

It is important to identify PM early. Early detection helps control the diseases through early intervention. In addition, for large farms, visual inspection takes long time, is labor intensive, and is costly. Moreover, the disease can be detected visually at middle to late stages of infection. This paper presents the use of UAV-based hyperspectral sensing and machine learning techniques for the detection and prediction of powdery mildew in grapes. Hyperspectral data collected from UAVs, digital images, visual inspection data, and proximal sensor data are used to develop machine learning models. The models are then used to detect the disease and disease severity.

One of the main advantages of machine learning technique is that they have potential to provide required information in real-time as UAVs are flying and collecting data, eliminating the need for post processing. This will be helpful in making immediate and real-time decisions such as application of fungicides immediately after the collection of the data from the UAVs. This reduces the turnaround time and helps reduce the impact of infection.

Data Collection and Processing

Cal Poly Pomona's existing vineyard, which is shown Figure 1, is being used for this research [Acosta et al., 2024]. At the early stages of PM infection, the disease is not usually visible on the canopy, and that is one of the challenges with the early detection of the disease using remote sensing technique.

Fig. 1. Cal Poly Pomona vineyard (left) and a PM infected grape leaves and fruits (right).

Remote sensing data was collected from a DJI Matrice 600 multicopter that is equipped with a co-aligned hyperspectral sensor from Headwall [Acosta et al., 2024] as shown in Fig. 2. The proximal sensor data collected include NDVI (normalized difference vegetation index) using FieldScout CM 1000 meter and chlorophyll content using a SPAD Plus Chlorophyll meter [Acosta et al., 2024].

Fig 2. Hyperspectral data collection of the vineyard from the UAV.

In addition, visual rating of disease severity was also collected as shown in Figure 3 [Acosta et al., 2024]. In the scale of 1-5, 1 is the least affected or diseased and 5 is the rating for the leaves with the most severe disease conditions. The visual rating data was also used for the machine learning model training as discussed later.

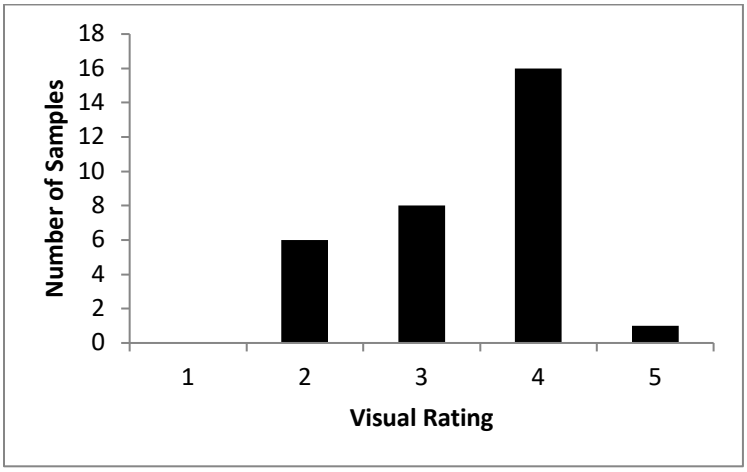

Fig 3. Visual rating of the disease severity.

Data Processing

The hyperspectral data from the co-aligned senor was processed using the AgView and Spectral View Software from Headwall [Acosta et al., 2024]. The processed remote sensing data was then used to calculate various vegetation indices including Red Edge Ratio, NDVI, modified NDVI (mND_{705}), photochemical reflectance index (PRI), and modified chlorophyll absorption ration index (MCARI). Figure 4 shows the correlation between WBI and NIR reading obtained from the NDVI meter as an example. A Pearson correlation coefficient (ρ) of -0.49 was obtained.

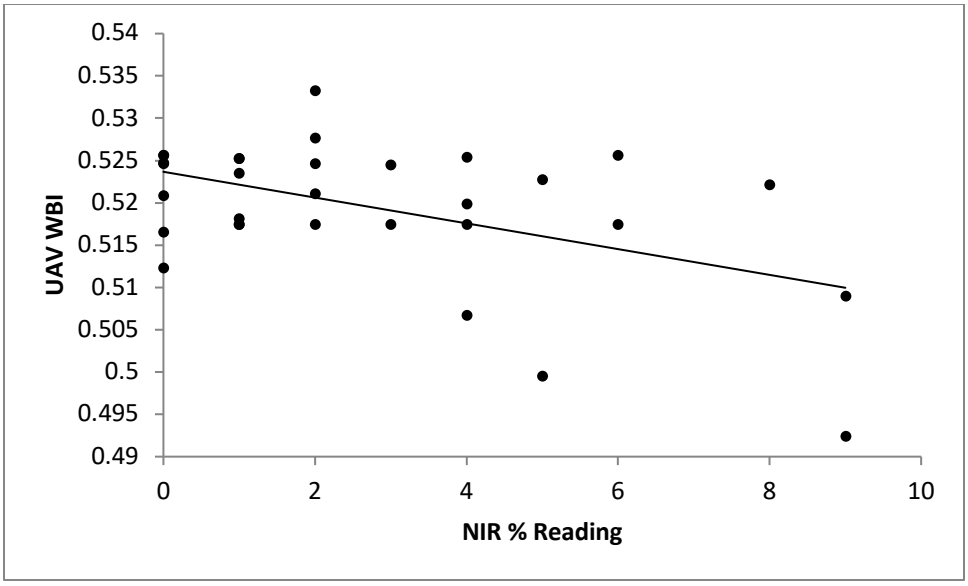

Fig 4. Relationship between UAV WBI and NIR reading (ρ = -0.49, p = 6 x 10^{-5}).

Machine Learning Pipeline

Vine Detection and Segmentation

The initial step involves using object detection techniques to detect grape vines on the collected images. These models, trained on a dataset of annotated images, accurately identify the presence and location of vines within the images. Then, the bounding boxes are fed into segmentation models, which are used as the prompt encoder to perform pixel-wise classification of the images, creating a detailed mask that highlights the exact regions occupied by the vines. This dual approach ensures comprehensive representation by providing both bounding boxes from object detection models and precise masks from segmentation models. Figure 5 shows a UAV image of the vineyard.

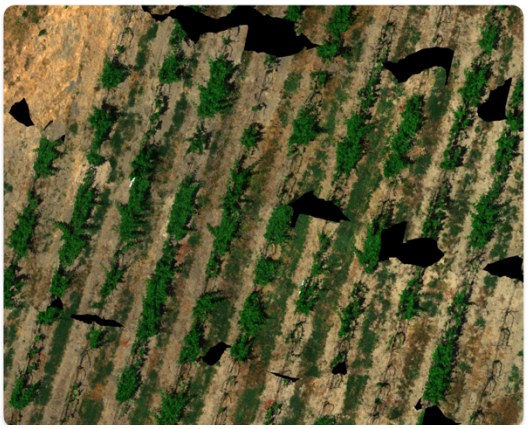

Fig. 5 UAV image of the vineyard.

Figure 6 on the left shows the detected vines in the image collected by the UAV. Figure 6 on the right shows the pixelwise classification of the vines.

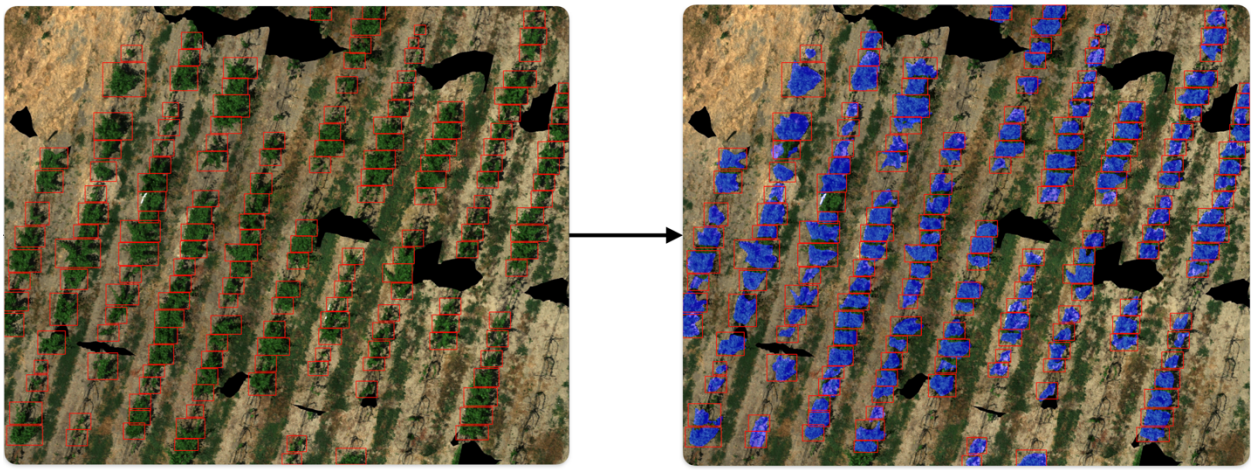

Fig. 6 Detected vines (left) and pixelwise classification of the images (right).

Bounding Box and Mask Extraction

Once the vines are detected and segmented, the next step involves extracting the bounding boxes produced by the object detection models along with the corresponding masks from the segmentation models. This process is critical for isolating the vines at a pixel level. Bounding boxes alone may include extraneous pixels from the ground or nearby vegetation, which could interfere with the accuracy of subsequent analyses. By combining the bounding boxes with precise masks, we ensure that only the pixels representing the vines are considered, enhancing the precision of our data. For the analysis in the next phase, only the union of the bounding boxes and their corresponding masks are considered.

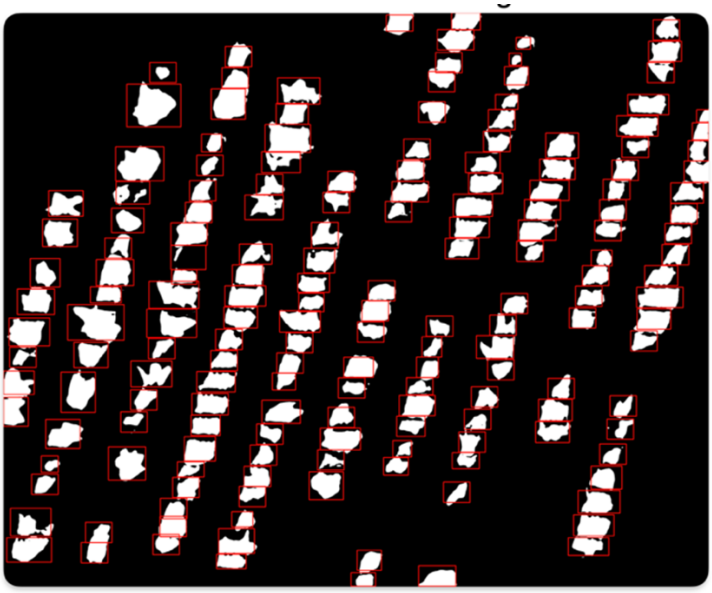

Fig. 7 Bounding boxes with their corresponding masks.

GPS locations of the sensors are used to map the collected data to the bounding boxes and masks of the vines detected by the developed models. This mapping process ensures that each vine's hyperspectral features are associated with accurate, real-world measurements, providing a reliable basis for model training and validation.

Training and Validation

In this phase, the available datapoints, both remote sensing and ground truth data, are utilized to ensure that our models are trained on high-quality, accurate data. This selective approach enhances the ability of the developed models to effectively detect and diagnose vine health

issues. Three different machine learning algorithms are used: a custom neural network, XGBoost, and a stacking algorithm. The data is divided into training and validation sets to evaluate the models' performance. During training, the models learn to identify and predict vine health indicators. Validation data set is used to assess the models' performances and make necessary adjustments to improve the accuracy and reliability of the developed models.

Once we have achieved good results on our validation sets, we can apply our method to detect powdery mildew and assess plant health in all the grapevines detected through our object detection and segmentation models. Using the trained and validated models, we process the images to identify and diagnose health issues in the vines. The combination of bounding boxes and masks ensures precise analysis, and the extracted hyperspectral features provide detailed insights into the health status of each vine. This method allows for large-scale monitoring and management of vineyard health, enabling timely interventions and improving overall grapevine productivity.

Machine Learning Models

Object Detection

We have utilized the YOLOv10 (You Only Look Once version 10) model and fine-tuned it on our drone imagery dataset to specifically detect vine trees. This customization ensures high accuracy in identifying and localizing vine trees within aerial images captured by our drones, enhancing the precision and reliability of our detection system.

The YOLOv10 architecture represents a significant advancement in real-time object detection. It incorporates various strategies to improve both efficiency and accuracy, addressing the issues found in previous YOLO versions. The architecture includes a backbone, a neck, and a head, with enhancements in each part to optimize performance.

The backbone of YOLOv10 uses a combination of convolutional layers to extract features from the input image. It leverages large-kernel convolutions and partial self-attention (PSA) modules to enhance the model's capability. The large-kernel convolution, specifically a 7×7 kernel, increases the receptive field, enabling better context capture for each pixel. The PSA module splits the feature map into two parts, processing only one part through a multi-head self-attention mechanism, which reduces computational cost while maintaining performance (Wang et al., 2024).

For the neck, YOLOv10 explores several feature fusion techniques such as PAN (Path Aggregation Network), BiC (Bidirectional Cross-scale), and GD (Global Dilated convolution). These techniques enhance multi-scale feature fusion, crucial for detecting objects of various sizes. Additionally, the architecture employs a spatial-channel decoupled downsampling method to reduce computational overhead. This method separates spatial reduction from channel increase operations, resulting in significant computational savings (Wang et al., 2024).

The head of YOLOv10 consists of a classification and regression part. It introduces a lightweight classification head that uses depthwise separable convolutions, significantly reducing the parameter count and computation. The dual label assignment strategy in the head ensures efficient and accurate bounding box predictions. During training, both one-to-many and one-to-

one label assignments are used, providing rich supervision and reducing the reliance on non-maximum suppression (NMS) during inference (Wang et al., 2024).

YOLOv10 uses several loss functions to train the model effectively. The Box Loss (L_box) is designed to measure the discrepancy between the predicted bounding boxes and the ground truth boxes. This loss function is defined as:

$$\mathbf{L_{box}} = \sum_{i=1}^{N} \boldsymbol{w_i} \, . \boldsymbol{IoU}(\boldsymbol{b_i}, \boldsymbol{b'_i}) \quad (1)$$

where N denotes the number of bounding boxes, $\boldsymbol{w_i}$ is a weight assigned to the i-th box, $\boldsymbol{b_i}$ represents the predicted bounding box, and $\boldsymbol{b'_i}$ is the corresponding ground truth box. The Intersection over Union (IoU) metric is used to evaluate the overlap between the predicted and ground truth boxes, ensuring that the model learns to produce precise bounding box coordinates (Wang et al., 2024; Zheng et al., 2019).

The Class Loss (L_cls) focuses on the accuracy of the predicted class labels. It is defined as:

$$\boldsymbol{L_{cls}} = \sum_{i=1}^{N} \sum_{c=1}^{C} -\boldsymbol{y_{ic}} \mathbf{log}\,(\boldsymbol{p_{ic}}) \quad (2)$$

In this equation, N is the number of instances, C is the number of classes, $\boldsymbol{y_{ic}}$ is a binary indicator (0 or 1) of whether class c is the correct classification for instance i, and $\boldsymbol{p_{ic}}$ is the predicted probability for class c. This loss function ensures that the model's predictions for class probabilities are accurate, encouraging the network to correctly classify objects within the image (Wang et al., 2024).

The Distribution Focal Loss (DFL) is introduced to address the imbalance in class distributions, giving more importance to hard-to-classify examples. It is formulated as:

$$\boldsymbol{L_{DFL}} = = \frac{-1}{N} \sum_{i=1}^{N} \sum_{c=1}^{C} -\boldsymbol{y_{ic}} \, \mathbf{log}(\mathbf{1} - \boldsymbol{p_{ic}})^{\gamma} \, \mathbf{log}\,(\boldsymbol{p_{ic}}) \quad (3)$$

where γ is a focusing parameter that controls the strength of the focal effect. By emphasizing the misclassified examples, this loss function helps the model to focus on learning difficult instances, thereby improving its overall robustness and accuracy (Wang et al., 2024; Li et al., 2020).

Lastly, the optimization process in YOLOv10 is driven by the Stochastic Gradient Descent (SGD) optimizer. The update rule for the SGD optimizer is:

$$\boldsymbol{\theta_{t+1}} = \boldsymbol{\theta_t} - \boldsymbol{\eta_t}(\boldsymbol{\nabla_\theta} \mathbf{L}(\boldsymbol{\theta_t}) + \boldsymbol{\lambda \theta_t}) \quad (4)$$

In this equation, $\boldsymbol{\theta_t}$ represents the model parameters at iteration t, $\boldsymbol{\eta_t}$ is the learning rate at iteration t, $\boldsymbol{\nabla_\theta} \mathbf{L}(\boldsymbol{\theta_t})$ is the gradient of the loss function with respect to the model parameters, and λ is the weight decay parameter. The weight decay term helps in regularizing the model by penalizing large weights, thereby reducing the risk of overfitting (Wang et al., 2024).

YOLOv10 combines advanced techniques to enhance performance and efficiency in real-time object detection. By fine-tuning YOLOv10 on our specific dataset, the model has learned the unique features of vine trees in aerial imagery, making our detection system robust and accurate.

Segmentation

In our work, we have incorporated the Segment Anything Model (SAM) as the backbone for segmentation tasks, enhancing our pipeline's efficiency and accuracy. The output of our object detection model is used as the input bounding boxes for SAM, which then performs pixel-wise classification within these bounding boxes to create detailed masks. This approach ensures that our segmentation process remains robust and responsive.

The architecture of the Segment Anything Model (SAM) consists of three main components: a Vision Transformer (ViT) backbone, a prompt encoder, and a mask decoder. The ViT backbone processes the input image to generate detailed feature maps by capturing long-range dependencies and high-level features. The prompt encoder takes user inputs, such as points, bounding boxes, or initial masks, and converts them into a format that informs the model about the object or region of interest. These encoded prompts, along with the feature maps, are then fed into the mask decoder, which iteratively refines the segmentation mask, aligning it precisely with the object's edges. This combination of a powerful feature extractor, flexible user input handling, and iterative mask refinement allows SAM to achieve high accuracy and adaptability across diverse segmentation tasks (Kirillov et al., 2023).

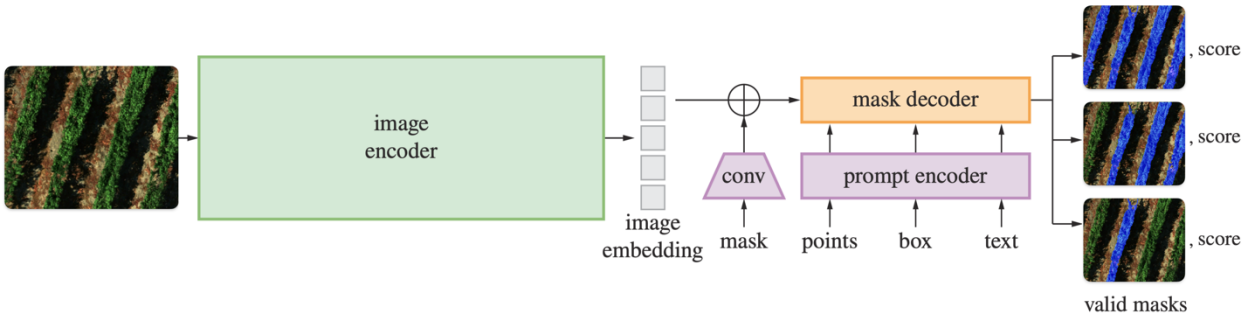

Fig. 8 Segmentation Anything Model (SAM) architecture.

The training algorithm for SAM involves simulating interactive segmentation with several key steps. Initially, a foreground point or bounding box is randomly selected for the target mask. Points are sampled uniformly from the ground truth mask, while boxes are perturbed with random noise. Subsequent points are then selected from the error region between the previous mask prediction and the ground truth mask. This process continues with up to eight iteratively sampled points, followed by two iterations without additional points. Losses are calculated after each iteration and backpropagated to update the model parameters (Kirillov et al., 2023).

SAM employs a combination of focal loss and dice loss in a 20:1 ratio, along with mean-square-error loss for Intersection over Union (IoU) prediction (Kirillov et al., 2023). The focal loss addresses class imbalance by focusing more on hard-to-classify examples, and is defined as:

$$\boldsymbol{Focal\ Loss} = -\boldsymbol{a_t}(\mathbf{1}-\boldsymbol{p_t})^{\gamma}\ \mathbf{log}\ (\boldsymbol{p_t}) \tag{5}$$

Where $\boldsymbol{p_t}$ is the model's estimated probability for the true class (Lin et al., 2018). The dice loss measures the overlap between the predicted and ground truth masks, and is defined as:

$$\boldsymbol{Dice\ Loss} = \mathbf{1} - \frac{\mathbf{2}|\boldsymbol{A}\cap\boldsymbol{B}|}{|\boldsymbol{A}|+|\boldsymbol{B}|} \tag{6}$$

where $\boldsymbol{A}$ is the set of predicted pixels, and $\boldsymbol{B}$ is the set of ground truth pixels (Sudre et al., 2017). The mean-square-error loss, used for the IoU prediction head, estimates the IoU between each predicted mask and the object it covers, and is defined as:

$$\boldsymbol{MSE\ Loss} = \frac{\mathbf{1}}{\boldsymbol{n}}\sum_{\boldsymbol{i}=\mathbf{1}}^{\boldsymbol{n}}(\boldsymbol{y_i}-\hat{\mathbf{y}}_{\boldsymbol{i}})^{\mathbf{2}} \tag{7}$$

where $\boldsymbol{y_i}$ is the ground truth IoU, and $\hat{\mathbf{y}}_{\boldsymbol{i}}$ is the predicted IoU (Kirillov et al., 2023).

By integrating SAM as the backbone for segmentation and using the bounding boxes from our object detection model, we have created a system capable of delivering precise segmentation

results with high accuracy. This approach ensures that our segmentation process is robust and reliable, making it suitable for applications requiring high precision.

Model Training

Three different approaches are used to learn the mapping of the features with ground truth data as discussed below.

Neural Networks

Neural networks are computational models inspired by the human brain's structure, comprising layers of interconnected neurons. The architecture typically includes an input layer, multiple hidden layers, and an output layer. Neurons apply nonlinear activation functions such as ReLU, sigmoid, or tanh. Training involves backpropagation to calculate gradients and gradient descent to optimize weights and biases, minimizing a specified loss function. These models are highly effective for complex pattern recognition tasks due to their ability to model nonlinear relationships (Schmidhuber, 2015).

In this case, a neural network with ReLU activation functions in the hidden layers and a softmax activation function in the output layer was used for predicting plant health. The ReLU (Rectified Linear Unit) activation function is defined as:

$$R(x) = \max(0, x) \tag{8}$$

where x is the input to a neuron. The softmax activation function is defined as:

$$softmax(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{N} e^{x}j} \tag{9}$$

where x_i is the input to the i-th neuron in the output layer and N is the number of output neurons.

XGBoost

XGBoost is an efficient and scalable machine learning algorithm based on gradient boosting. It builds an ensemble of weak learners, typically decision trees, in a sequential manner where each tree corrects the residual errors of the previous ones by optimizing a differentiable loss function through gradient descent. The iterations can be shown in the following formula:

$$\hat{y}_i^{(t)} = \hat{y}_i^{(t-1)} + f_t(x_i) \tag{10}$$

Where f_t is the decision tree at the t-th iteration. XGBoost incorporates both L1 (Lasso) and L2 (Ridge) regularization to control overfitting and complexity of the models. α is used for L1 with a default of 0, and λ is for L2 with a default of 1. It efficiently handles sparse data and missing values and supports parallel processing, enhancing computational performance and scalability. The algorithm can also prune the trees to avoid overfitting (Chen & Guestrin, 2016).

Stacking

Stacking, or stacked generalization, is an ensemble learning method that integrates multiple models to improve predictive performance. It involves training several base models (level-0) on a dataset and then using their outputs as input features for a second-level model (meta-learner or level-1). The meta-learner is trained to optimize the combination of base model predictions. This method typically employs cross-validation to prevent overfitting and ensures the meta-learner effectively generalizes from the base model outputs, leveraging their diverse strengths.

In this approach, three base learners, including Support Vector Machine (SVM), Random Forest, and Extreme Gradient Boosting (XGB), were used. The meta-learner was logistic regression.

Results and Discussion

In this study, we evaluated the performance of three different machine learning models, Neural Network, XGBoost, and a Stacking Model on predicting plant health. The models were assessed based on their accuracy, recall, precision, and F1-score on the testing dataset.

Table 1. Precision, recall, and F1 score of the models on the testing dataset.

Model	Precision	Recall	F1 Score
Neural Network	0.8750	1.0000	0.9333
XGboost	0.8571	0.8571	0.8571
Stacking	0.6049	0.7778	0.6806

Figure 9 shows the training and test accuracy of the three models.

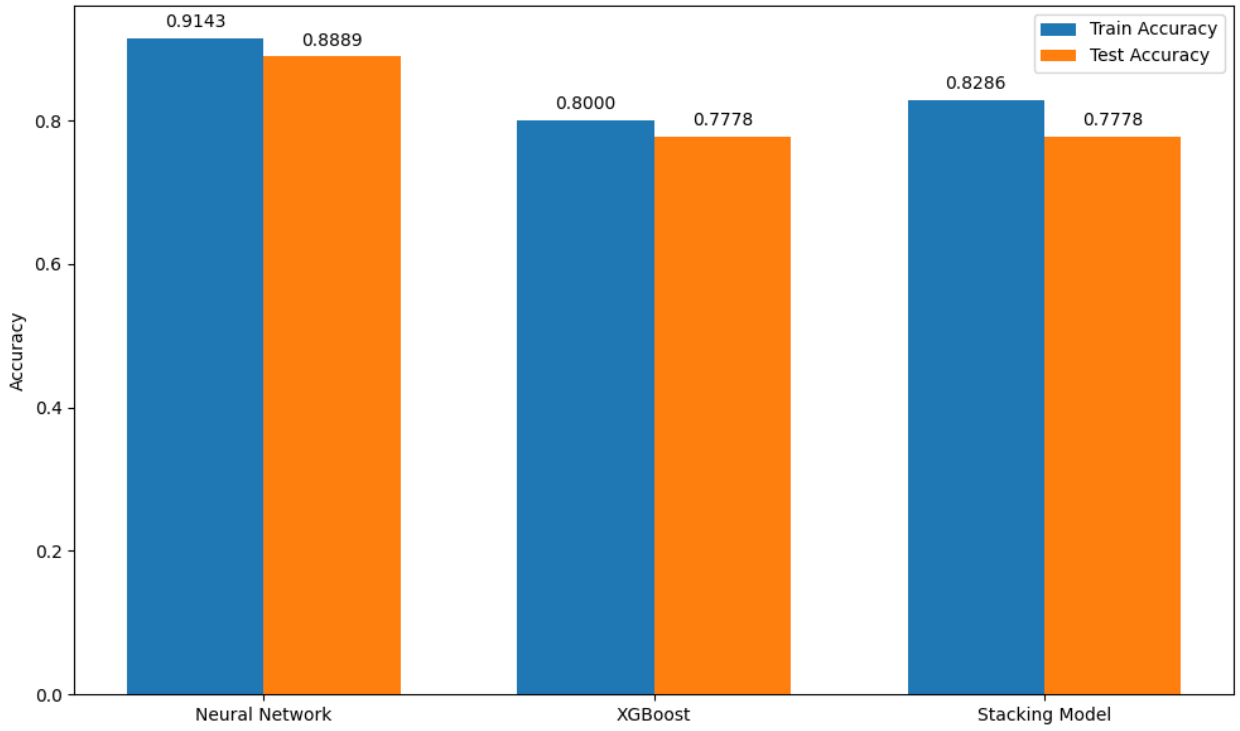

Fig. 9 Train and test accuracy comparison for Neural Network, XGBoost, and Stacking Models.

The Neural Network achieved a training accuracy of 91.43% and a testing accuracy of 88.89%. This high level of performance indicates that the model learned the training data well and generalized effectively to unseen data. With a precision of 0.8750 and a perfect recall of 1.0000, the Neural Network successfully identified all relevant instances while maintaining a high rate of correct positive predictions. The resulting F1 score of 0.9333, which balances precision and recall, further validates the model's robustness and strong performance.

The XGBoost model demonstrated consistent performance, with a training accuracy of 80.00% and a testing accuracy of 77.78%. Its balanced precision and recall, both at 0.8571, imply that the model is effective at identifying positive instances while maintaining a moderate rate of false positives. The F1 score of 0.8571 reflects this balance, making XGBoost a reliable model, though it does not reach the same level of accuracy or F1 score as the Neural Network. This consistency suggests that while XGBoost captures some complexity in the data, it is not as effective as the Neural Network.

The Stacking Model, which combines multiple base learners, achieved a training accuracy of

82.86%, but saw a slight decrease in testing accuracy of 77.78%. The precision of 0.6049 indicates a higher rate of false positives compared to the other models. While the recall of 0.7778 is comparable to the testing accuracy, it shows that the model identifies positive instances reasonably well but struggles with precision. The F1 score of 0.6806 reflects these challenges and suggests that the Stacking Model may require further tuning or different base learners to improve its precision and overall performance.

Comparatively, the Neural Network stands out as the most effective model in this study, as evidenced by its highest training and testing accuracies, perfect recall, and the highest F1 score. This model's performance indicates that it can generalize well from training data to unseen data without overfitting as the difference between its training and testing accuracy is minimal. Although XGBoost performs consistently, with balanced precision and recall, it does not reach the same level of accuracy or F1 score as the Neural Network. Its lower training and testing accuracies suggest it might not capture the complexity of the data as effectively as the Neural Network.

The Stacking Model, while incorporating multiple learners, shows promise with its intermediate training accuracy but requires further optimization. The lower precision and F1 score indicate that it needs improvement in managing false positives. The observed overfitting, where the training accuracy exceeds the testing accuracy by a notable margin, suggests that the model could benefit from additional tuning to enhance its generalization capabilities.

It is important to note that our models were trained on a relatively small dataset. To strengthen our models and improve their performance, we plan to gather additional data in future research. This will help address issues of overfitting and underfitting, thereby enhancing the overall accuracy and robustness of our models.

Additionally, to better understand the relationships and potential collinearities among the features used in our models, we have provided a correlation matrix below. This matrix helps in identifying which features have strong linear relationships and can inform feature selection and engineering efforts in future iterations of this work.

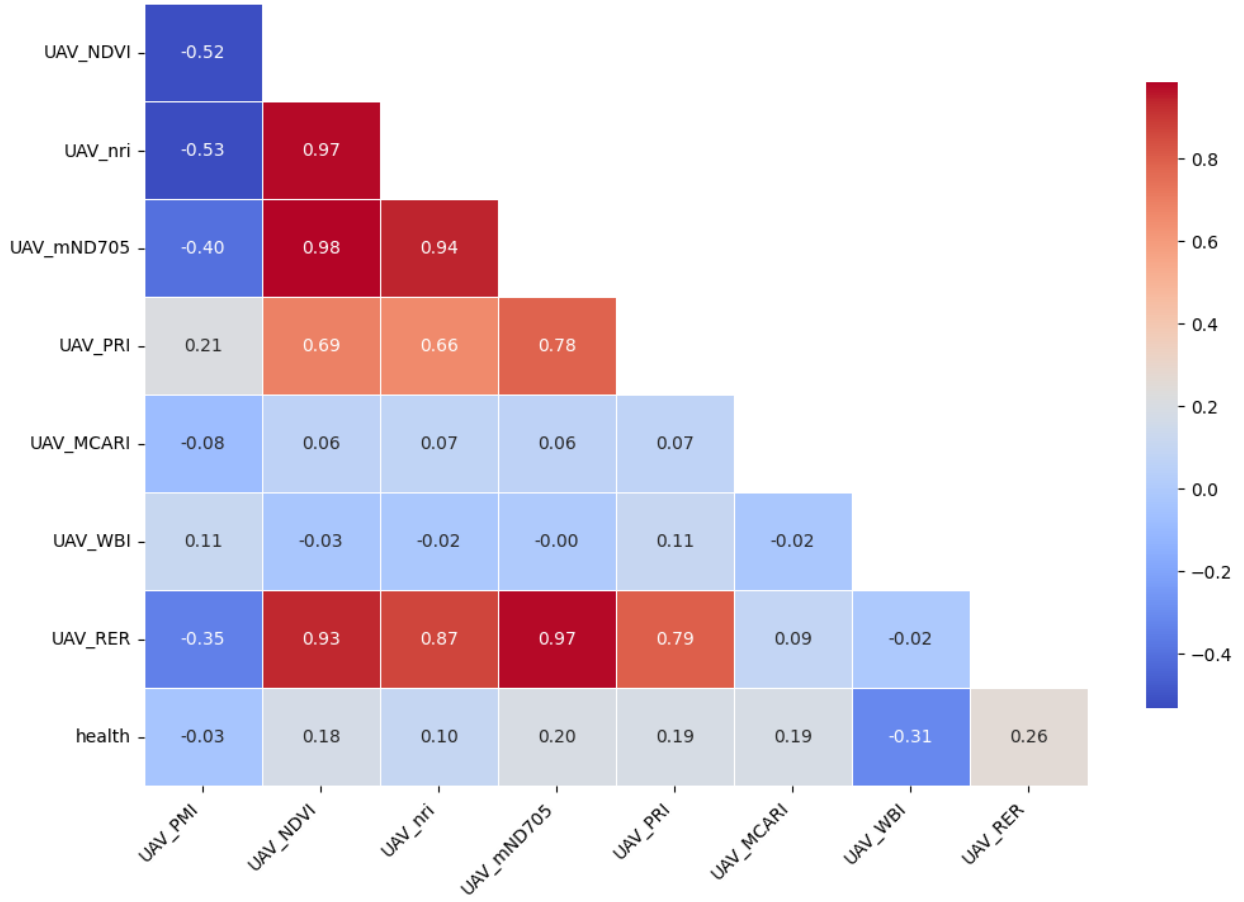

Fig. 10 Correlation heatmap of various UAV indices and plant health.

It is important to note that our models were trained on a relatively small dataset. To strengthen our models and improve their performances, we plan to gather additional data in future research. This will help address issues of overfitting and underfitting, thereby enhancing the overall accuracy and robustness of our models.

Conclusion and Future Work

In this study, we implemented a comprehensive machine learning pipeline to evaluate grapevine health using UAV-derived indices and hyperspectral features. The process began with vine detection and segmentation, combining object detection and segmentation models to accurately isolate vine pixels. This approach ensured precise analysis by extracting bounding boxes and masks for each vine, mapped with GPS data to associate real-world measurements.

Three models, Neural Network, XGBoost, and a Stacking Model, were trained and validated using high-quality data. The Neural Network outperformed others, demonstrating strong generalization capabilities. XGBoost showed balanced performance, while the Stacking Model required further optimization.

Future work will focus on improving the robustness of the Stacking Model, exploring other ensemble techniques, and leveraging larger datasets to further enhance predictive performance. By addressing these areas, we aim to develop even more accurate and reliable models for large-scale vineyard health monitoring, enabling timely interventions and improving overall grapevine productivity.

Acknowledgments

The authors would like to acknowledge the support from California State University's Agricultural

Research Institute (ARI). The project was supported by the ARI Grant Number 23-04-113.

References

Acosta, M., Pena, J., Sherafat, A., Gonzalez, C., Sherman, T., Bhandari, S., and Raheja, A. (2024). Investigating the Potential of UAV-Based Hyperspectral Sensor in Detecting Powdery Mildew in Grapes. SPIE Defense + Commercial Sensing, Proceedings of Autonomous Air and Ground Sensing Systems for Agricultural Optimization and Phenotyping III, Kissimmee, FL.

Chen, T., & Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM. https://doi.org/10.1145/2939672.2939785

Kirillov, A., Mintun, E., Ravi, N., Mao, H., Rolland, C., Gustafson, L., Xiao, T., Whitehead, S., Berg, A. C., Lo, W.-Y., Dollár, P., & Girshick, R. (2023). Segment Anything. arXiv. https://arxiv.org/abs/2304.02643

Li, X., Wang, W., Wu, L., Chen, S., Hu, X., Li, J., Tang, J., & Yang, J. (2020). Generalized Focal Loss: Learning Qualified and Distributed Bounding Boxes for Dense Object Detection. arXiv. https://arxiv.org/abs/2006.04388

Lin, T.-Y., Goyal, P., Girshick, R., He, K., & Dollár, P. (2018). Focal Loss for Dense Object Detection. arXiv. https://arxiv.org/abs/1708.02002

Schmidhuber, J. (2015). Deep learning in neural networks: An overview. Neural Networks, 61, 85-117. Elsevier BV. https://doi.org/10.1016/j.neunet.2014.09.003

Sudre, C. H., Li, W., Vercauteren, T., Ourselin, S., & Cardoso, J. M. (2017). Generalised Dice Overlap as a Deep Learning Loss Function for Highly Unbalanced Segmentations. In Lecture Notes in Computer Science (pp. 240-248). Springer International Publishing. https://doi.org/10.1007/978-3-319-67558-9_28

Wang, A., Chen, H., Liu, L., Chen, K., Lin, Z., Han, J., & Ding, G. (2024). YOLOv10: Real-Time End-to-End Object Detection. arXiv. https://arxiv.org/abs/2405.14458

Zheng, Z., Wang, P., Liu, W., Li, J., Ye, R., & Ren, D. (2019). Distance-IoU Loss: Faster and Better Learning for Bounding Box Regression. arXiv. https://arxiv.org/abs/1911.08287