Monitoring Vine Plant Health with a Classification Model

Emmanuel Piazza, Edoardo Focacci, Antonino Patania, Salvatore Patisso  
 [e.piazza3@studenti.unipi.it](mailto:e.piazza3@studenti.unipi.it), [e.focacci@studenti.unipi.it](mailto:e.focacci@studenti.unipi.it), [a.patan](mailto:a.patan)ia4@studenti.unipi.it, [s.patisso@studenti.unipi.it](mailto:email@email.com)

ABSTRACT

This application is designed to monitor the health of grapevine plants by capturing images of their leaves and analyzing them through a classifier. The classifier determines whether the plant is healthy or affected by a specific disease. The application offers a comprehensive solution for vineyard management by providing multiple functionalities. Users can take photos of grapevine leaves, which are then processed by an integrated classifier that uses machine learning algorithms to identify potential diseases. The results are promptly displayed, indicating either a healthy state or the presence of a disease.

In addition to diagnostic capabilities, the application features a gallery where users can view all captured photos. This allows for easy tracking and comparison of plant health over time. Furthermore, the application includes a mapping function that displays the geographical locations of the captured images. This feature is particularly useful for identifying and managing disease outbreaks across different sections of the vineyard.

By combining image capture, disease classification, and geolocation mapping, this application serves as a powerful tool for vineyard owners and managers. It enhances the efficiency of monitoring plant health, facilitates early disease detection, and supports targeted intervention strategies, ultimately contributing to healthier crops and increased productivity.

1 Introduction

In the viticulture industry, maintaining the health of grapevine plants is crucial for ensuring high-quality yields. Grapevines are susceptible to a variety of diseases that can significantly impact both the quality and quantity of the harvest. Traditional methods of monitoring plant health often rely on manual inspection and expertise, which can be time-consuming and prone to human error.

The problem addressed by this application is the need for a more efficient, accurate, and accessible way to monitor the health of grapevine plants. Current approaches typically involve periodic manual checks, which may not provide timely detection of diseases. Additionally, many existing digital solutions lack integration between diagnostic capabilities and practical management tools such as mapping and historical tracking.

This application distinguishes itself by integrating machine learning techniques for disease classification with user-friendly features for image management and geolocation. The combination of capturing images, classifying plant health, and mapping the locations of affected plants offers a comprehensive solution that enhances decision-making processes and supports proactive disease management. This approach not only saves time and reduces labor but also improves the overall health and productivity of the vineyard.

2 Architecture

The architecture of the application is designed to provide seamless integration of image capture, disease classification, and geolocation mapping functionalities.

2.1 Front-end (mobile application)

The application is composed of a *Main Activity* element that manages the *Camera button* functionality and various permissions, each of the application other main features are divided into different pages. Each page is included inside a *Fragment* element, accessible by the user by using the sidebar.

**Image Capture**: Users can take high-quality photos (by using the built-in camera) of grapevine leaves, images are saved inside the device locale storage. The classification of the plant photo is done just after the photo has been taken. Additional information of the image (like timestamp and plant status) is saved by using the *exIfInterface* library. Each information is assigned to a *tag* during the saving phase and retrieved, when necessary, by using the same assigned *tag*.

**User Interface**: A user-friendly interface allows users to easily navigate through the app, view captured images, access diagnostic results and visualize all plant information inside the map.

**Geolocation**: The app uses GPS to tag the location where each photo is taken, a timestamp is also saved in order to create a history of the plant status.

**Map**: The use of a map allows the user to keep track of his plants located throughout the territory.  
Through the click of a marker it is possible to get some information of the plant.  
This has been made possible by the OpenStreetMap open source database and the use of the android library 'osmdroid' which allows for easy and intuitive interaction with the map.

2.2 Classification model

In order to classify correctly the various diseases that affect vine plants, we decided to adopt a CNN. There are numerous reasons for this decision. Here are some:

CNNs excel at detecting important features without needing any manual feature extraction. The layered architecture also helps in capturing different details; one of the entry layers can be used to recognize the edges of a leaf, and subsequent layers can become more specialized in detecting diseases. CNNs are also robust in varying image orientations, sizes, and lighting conditions; a photo can be taken with these various features changing from one instance to another. Additionally, the adaptability of CNNs makes them the ideal candidate for a mobile app

Immagine che contiene testo, schermata, diagramma, Carattere

Descrizione generata automaticamente

**Figure 1: An example of a CNN**

To develop an accurate CNN model, I’ll refer to this paper: plant\_disease\_classifier/report.pdf at main · Froffri/plant\_disease\_classifier (github.com).

And this dataset, publicly available from Kaggle, was used: [Augmented Grapevine Disease Dataset (kaggle.com)](https://www.kaggle.com/datasets/rm1000/augmented-grape-disease-detection-dataset)

2.3 Model tuning

Whereas all analyses were conducted on a different dataset, another great feature of CNNs is that a pre-trained model can be fine-tuned to develop a more specialized model. The dataset contains 1000 images for each class, with one class being "Healthy," and the other three representing common diseases for vine plants ("Black Rot," "Esca," and "Leaf Blight").

To implement a CNN, TensorFlow and Keras were used for the purpose of image classification. Two different models were built, and the one with higher accuracy on the test set was selected (one from scratch and one using a pre-trained CNN), and later compared. The two different models are "from-scratch," where no pre-trained weights were used and all layers were customizable, and one where an initial architecture based on VGG16 was used, with some personalized layers added. Differently from the paper, the two models presented similar accuracy, and we decided to proceed with the "from scratch" model trained over 10 epochs. Here below are the accuracy and loss function results over 10 epochs.

The model architecture is structured as follows: There are two layers of data augmentation, used to enhance the dataset and artificially "increase" the number of examples seen, and rescaling, used to normalize all the pixels on a scale from 0 to 1. Then, multiple convolutional layers with varying numbers of filters, kernel sizes, and activation functions (ReLU) are introduced to extract features from the input images. Max-pooling and batch normalization layers are applied subsequently to downsample and normalize the feature maps. Three more layers are added: the Flatten layer, Dropout layer, and Dense layer. These layers are required to convert the output of the convolutional layers to a 1D vector, deactivate a portion of the input image (used against overfitting), and one Dense Layer with the same number of neurons as the classes (4) and a Softmax activation function.

3 Experimental results

The model was compiled using these settings:

• L**oss**: we used categorical\_crossentropy because it quantifies the dissimilarity between predicted class probabilities and actual class labels.

The formula for the categorical crossentropy is the following:

where is the true probability distribution for class i and is the predicted probability for class i as output by the model.

• **Optimizer**: the algorithm used to update the model’s weights during training is Adam.

• **Metrics**: we chose to use accuracy as our metric due to the balanced dataset, its interpretability, and the absence of a need to favor either FP or FN

The model is trained using the *model.fit()* method with the training and validation datasets. Training spans 10 epochs.

Immagine che contiene testo, linea, Diagramma, diagramma

Descrizione generata automaticamente

Immagine che contiene linea, testo, Diagramma, diagramma

Descrizione generata automaticamente

**Figure 2: Results over Epochs**

3.1 Conversion and usage on mobile applications

As stated previously the model was built using Keras and Tensorflow, in order to not drain excessive resources on the mobile application the model was converted using the TensorFlow Lite Converter to transform it into the optimized TFLite format.

3.2 Experimental conclusions

The experimental results from our convolutional neural network (CNN) designed for grapevine disease detection in a mobile application show promising performance, particularly in the ability of the model to quickly learn from the training data. As can be observed from the graphs, the learning curve flattens early, indicating that the CNN rapidly assimilates the training patterns. This is further evidenced by achieving high accuracy rates exceeding 97% on previously unseen images.

However, when testing with other real-world images, signs of overfitting are observed concerning disease presence. This phenomenon is likely due to the balanced nature of our dataset, which may not fully represent the true distribution of disease occurrences in a natural environment. The high performance on the test dataset contrasts with the more mixed results on real images, highlighting the challenge of overfitting in practical applications. This suggests the need for further refinement of the model and perhaps an enrichment of the dataset with a more varied set of images to better generalize the CNN’s diagnostic capabilities across more diverse conditions.

4 Conclusion and future improvements

This application provides a tool for monitoring grapevine health through image capture, machine learning-based disease classification, and geolocation mapping. The high accuracy of the convolutional neural network (CNN) in identifying common grapevine diseases supports timely and targeted interventions, enhancing crop health and yield quality.

Despite its promising performance, the model shows signs of overfitting when tested with other real-world images. This indicates the need for further refinement and a more diverse dataset to improve the model’s generalization to varied conditions.

Future work will focus on:

**- Dataset Expansion**: Gathering a wider range of images to better represent real-world variability. If the app targets specific market segments, the dataset can be adjusted to be balanced to counteract overfitting.

- **User Experience**: Enhancing the app’s interface and adding features like disease information and treatment suggestions.

- **Disease Solutions**: Including potential solutions for detected diseases, providing users with actionable recommendations.

REFERENCES

[Augmented Grapevine Disease Dataset (kaggle.com)](https://www.kaggle.com/datasets/rm1000/augmented-grape-disease-detection-dataset)

plant\_disease\_classifier/report.pdf at main · Froffri/plant\_disease\_classifier

￼