🍇 PERNOSPHERE

Introduction:

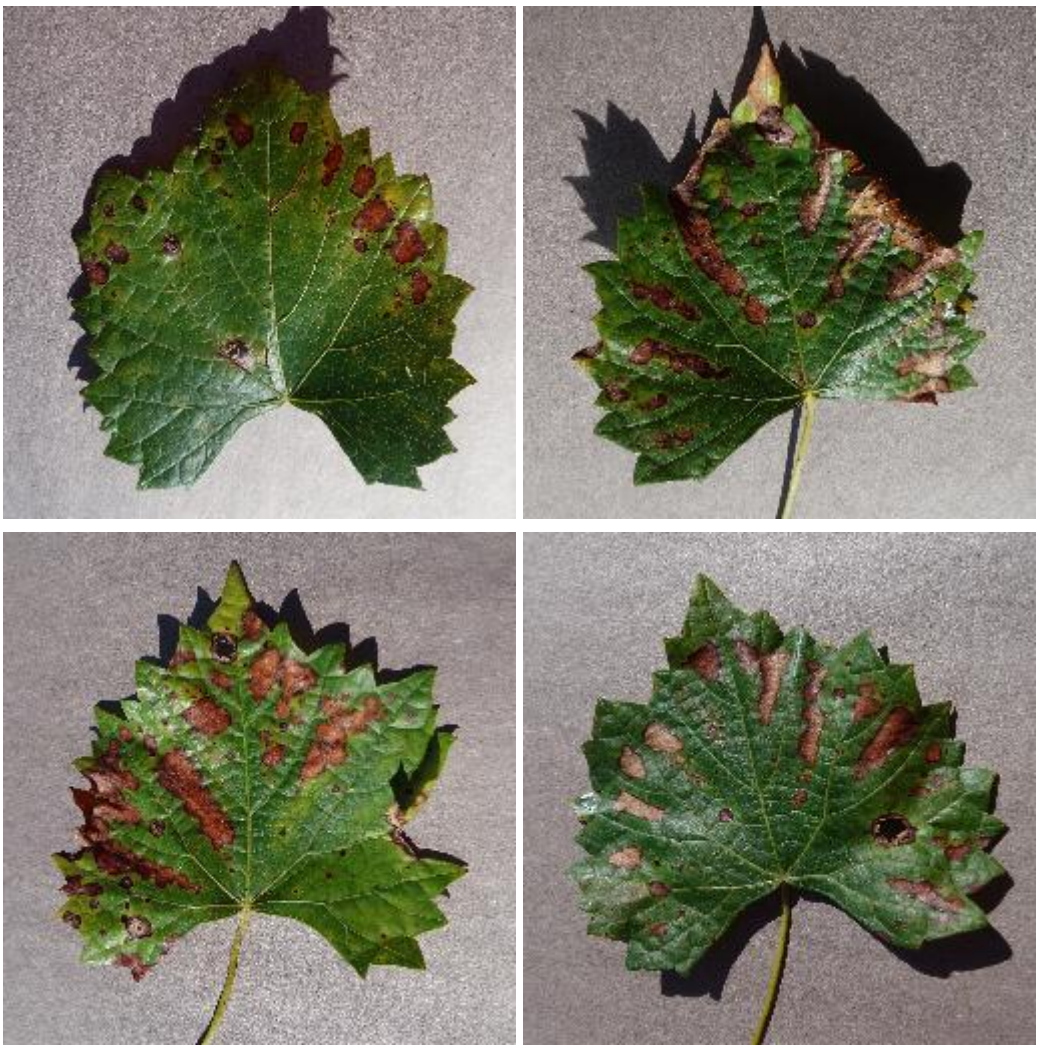
I have decided to focus on the topic of diseases that can affect grapevines. These may all look the same to an untrained eye, but in reality, they have distinctive characteristics that allow them to be identified. Below, I describe some of the main conditions that can affect grapevine leaves:

Black Rot:

Black Rot is a fungal disease caused by the pathogen *Guignardia bidwellii*. It initially appears as small brown or black spots on the leaves, which expand into circular lesions with a dark border and a light brown center. If left untreated, it can spread to the grape clusters, causing them to completely shrivel. It thrives in warm and humid conditions.

ESCA:

Esca is a complex wood disease caused by various fungi (such as *Phaeomoniella chlamydospora* and *Phaeoacremonium spp.*). Leaf symptoms include chlorotic (yellowish) and necrotic streaks, with a distinctive “tiger stripe” pattern due to the irregular arrangement of the lesions.



Leaf Blight:

Leaf Blight is a condition that can be caused by various pathogens or environmental stress. It manifests as drying and browning of the leaf blades, often starting from the edges. Affected leaves look burnt, curled, and may fall off prematurely.

Obviously, these conditions require different treatments, and it is important to be able to recognize them. For this reason, I have decided to apply computer vision to this field.


An healthy leaf has this aspect:




The DATASET and the MODEL

I used a Kaggle free and public database

link:
https://www.kaggle.com/datasets/rm1000/augmented-grape-disease-detection-dataset

And I decided to classify those with CNN.
I used a CNN because

A CNN automatically learns features like edges, shapes, and textures.

It requires fewer parameters than fully connected networks, making training faster.

It is translation-invariant, so they can recognize patterns even if the image is shifted or rotated.

My model is divided in two main parts:

1) **Feature extractor:**

Divided in 4 layers, it extracts features from the photos, to perform features extraction I used:

a) Conv2d: Extracts patterns like edges or textures.

b) BatchNorm2d: Normalizes the activations for more stable and faster training.

c) ReLU: Adds non-linearity.

d) MaxPool2d: Reduces the spatial size, keeping only the most important features.

2) **Classifier (Fully Connected layers):**

After extracting features, the network flattens the feature maps and passes them through an MLP (Multi-Layer Perceptron) to perform classification:

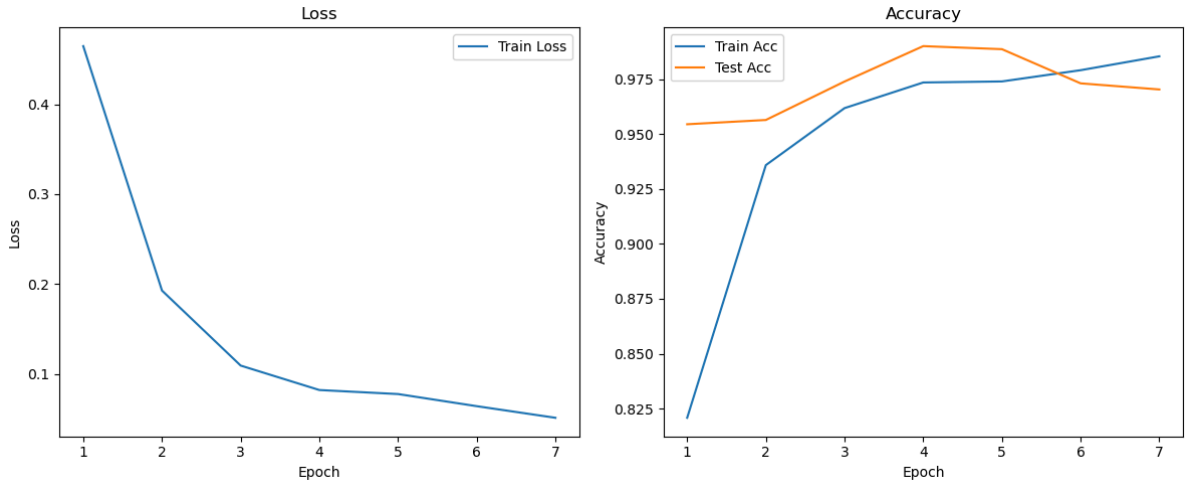
- Linear(9216, 512) → Linear(512, 128) → Linear(128, 4): These layers progressively reduce the dimensionality and eventually map to 4 output classes (Black Rot, ESCA, Healthy, Leaf Blight).
- Dropout(0.5): This function disables neurons during training, this avoid overfitting
- ReLU: Activates neurons non-linearly to learn complex relationships.

Results:

To avoid overfitting and other issues during training, I implemented several strategies in the train_test.py script:

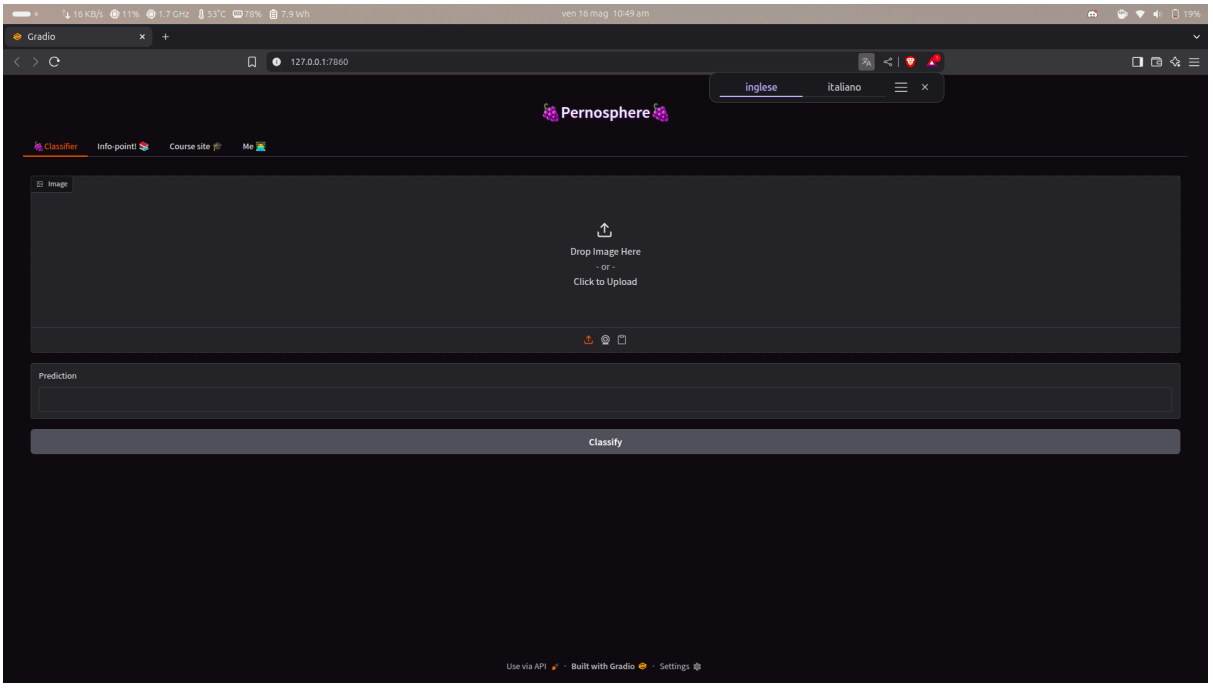
- Early Stopping: if the test accuracy doesn't increase for 3 epochs the program saves the model and terminates the training. That avoids overfitting.
- Dropout in the model: A dropout layer (0.5) is included in the MLP to randomly deactivate neurons during training, helping generalization and avoid the overfitting.
- GPU/Accelerator Detection: A line that detects the best device where to run the model. That avoids spending time with a slower device.

With this setup I obtained these results:



GUI:

The GUI is designed to interact with the program and includes a built-in wiki as well as links to the course materials.



REAL LIFE TEST: