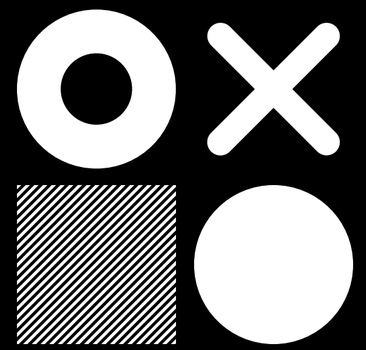
AI project

Barbato Daniele - 883661
Bologa Nicolae - 901121
Facini Antonella - 900455
Nica Sergiu - 874965

PLANT DESEASE DETECTOR



Custom CNN from scratch

CNN

Since the goal of our project is to identify what disease a plant has starting from an image, we used a CNN, which is particularly useful for finding patterns in images in order to recognize objects, classes, and categories

Features:

Automatic Feature Extraction

- CNNs automatically learn relevant patterns (such as edges, shapes, and textures) from images.
- This is especially useful for detecting visual symptoms of plant diseases.

Layered Architecture:

- CNNs are composed of several types of layers:
  - Convolutional Layers: extract local features.
  - ReLU (Activation Layers): introduce non-linearity.
  - Pooling Layers: reduce dimensionality and noise.
  - Fully Connected Layers: perform final classification.

Weight Sharing:

- Filters (kernels) are shared across the image, significantly reducing the number of parameters.
- This makes CNNs efficient and scalable

Spatial invarinace:

- CNNs are robust to translations, rotations, and scale variations.
- They can recognize a disease even if the leaf is tilted or partially visible.

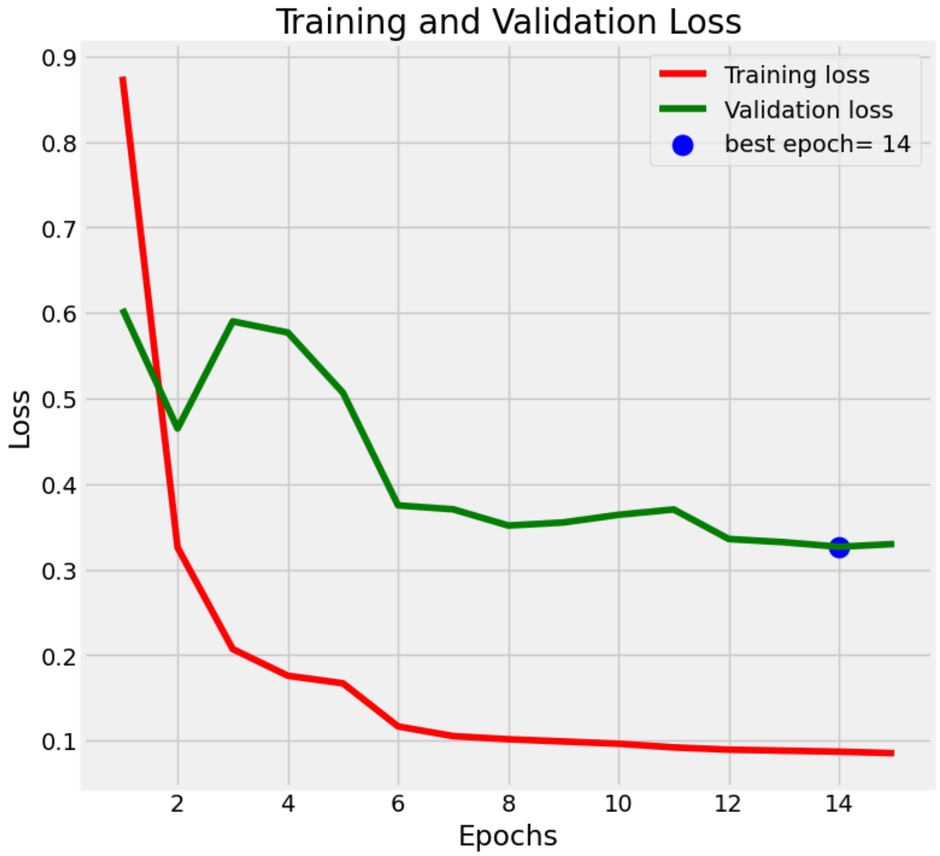
Hierarchical Feature Learning:

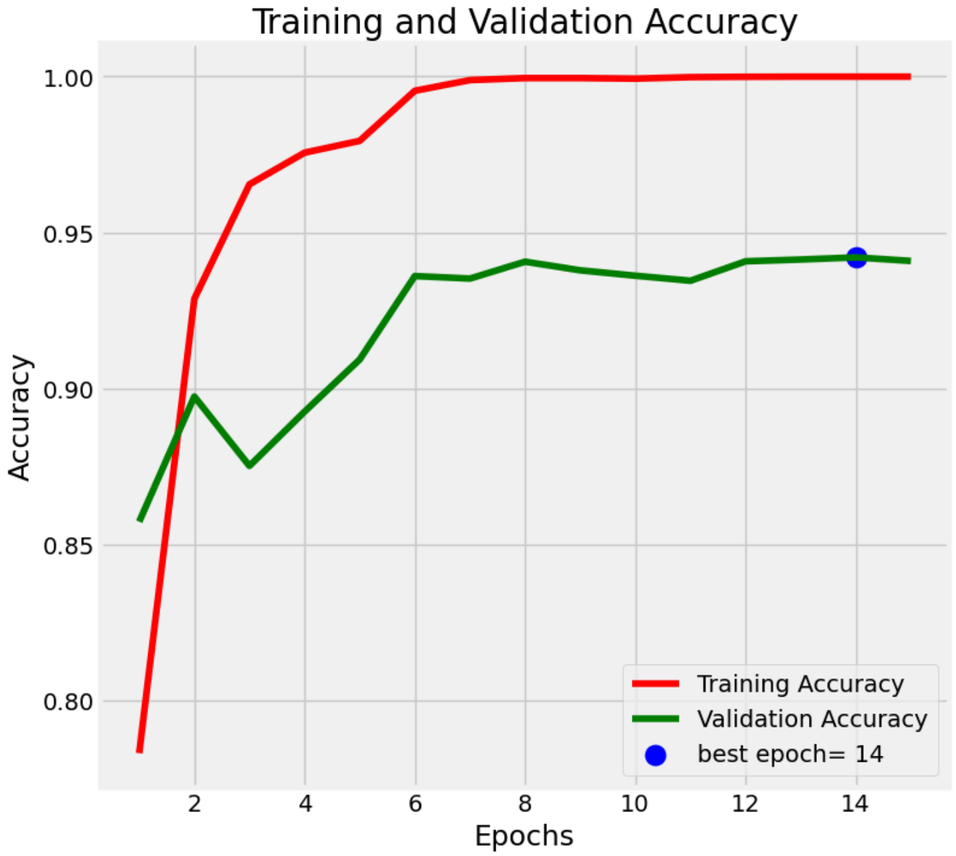
- Early layers detect simple patterns (e.g., edges), while deeper layers recognize complex structures (e.g., spots, mold, deformations).
- This hierarchical approach is ideal for analyzing natural images.

Generalization Capability:

- When properly trained, CNNs can generalize to new, unseen images that are similar to the training data.

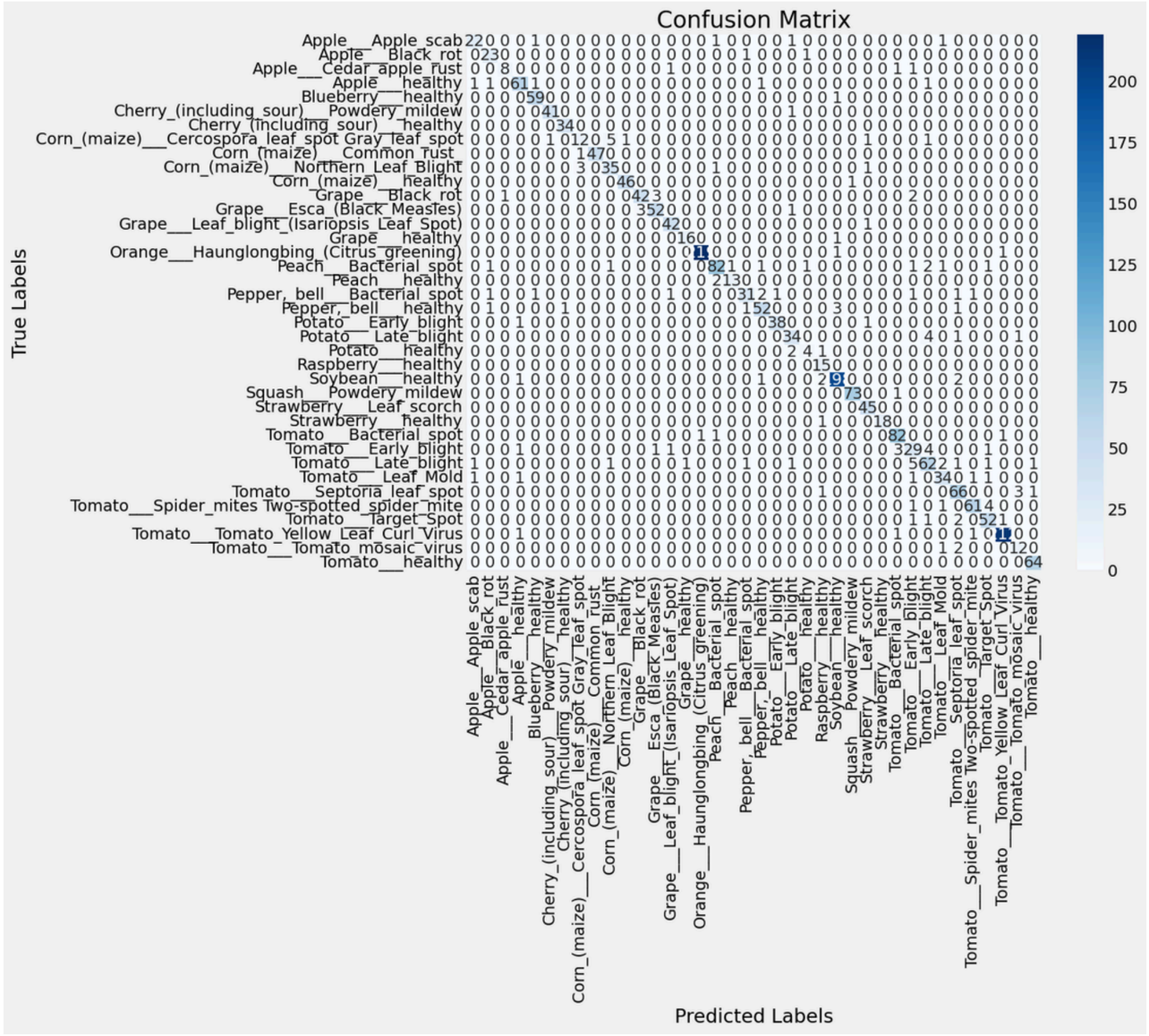
Training and Validation Accuracy and Loss





The model showed strong learning progress, reaching its highest accuracy and lowest validation loss at epoch 14, indicating effective training and generalization

Confusion Matrix



The confusion matrix shows how well a classification model performed across multiple plant disease categories. Here's a concise description:

- Axes: True labels (y-axis) vs. Predicted labels (x-axis).
- Diagonal cells: Correct predictions — these are mostly dark, indicating high accuracy.
- Off-diagonal cells: Misclassifications — lighter and fewer, suggesting low error rates.
- Overall: The model performs well, with most predictions correctly matching the true labels.

The custom CNN shows a good generalization with a 94% accuracy. by looking at the classification report some classes display a lower recall or F1-score(f.e. potato_healthy or tomato_early_blight), which might be caused by the smaller support (number of samples). In general: most classes exhibit strong performance; the model distinguishes well between healthy and deseased Leaves; no cases of zero-precision

Pre-trained algorythm: ResNet50

ResNet50

Residual Neural Netwok with 50 layers

ResNet50 is a deep convolutional neural (CNN) network architecture designed to reduce the vanishing gradient problem in deep neural networks by introducing residual connections.
This model was implemented using the PyTorch deep learning framework, leveraging its flexibility and dynamic computation graph to build and train a ResNet50 model.

Features:

Residual Blocks:

- Instead of learning a direct mapping, ResNet50 learns residual functions (the difference between input and output)
- Each residual block contains shortcut connections that allow gradients to flow directly through the network, improving training efficiency.

Bottleneck Architecture:

- ResNet50 uses a 1x1 → 3x3 → 1x1 convolution structure within each residual block to reduce computational cost while maintaining depth.
- The 1x1 convolutions are used for dimensionality reduction and expansion

50-layer structure:

- The network consists of 49 convolutional layers + 1 fully connected layer
- Breakdown:
  - Conv1: 7x7 convolution (64 filters) + max pooling.
  - Conv2_x: 3 residual blocks (each with 64, 64, 256 filters).
  - Conv3_x: 4 residual blocks (each with 128, 128, 512 filters).
  - Conv4_x: 6 residual blocks (each with 256, 256, 1024 filters).
  - Conv5_x: 3 residual blocks (each with 512, 512, 2048 filters).
  - Final Layers: Average pooling + fully connected (for classification).

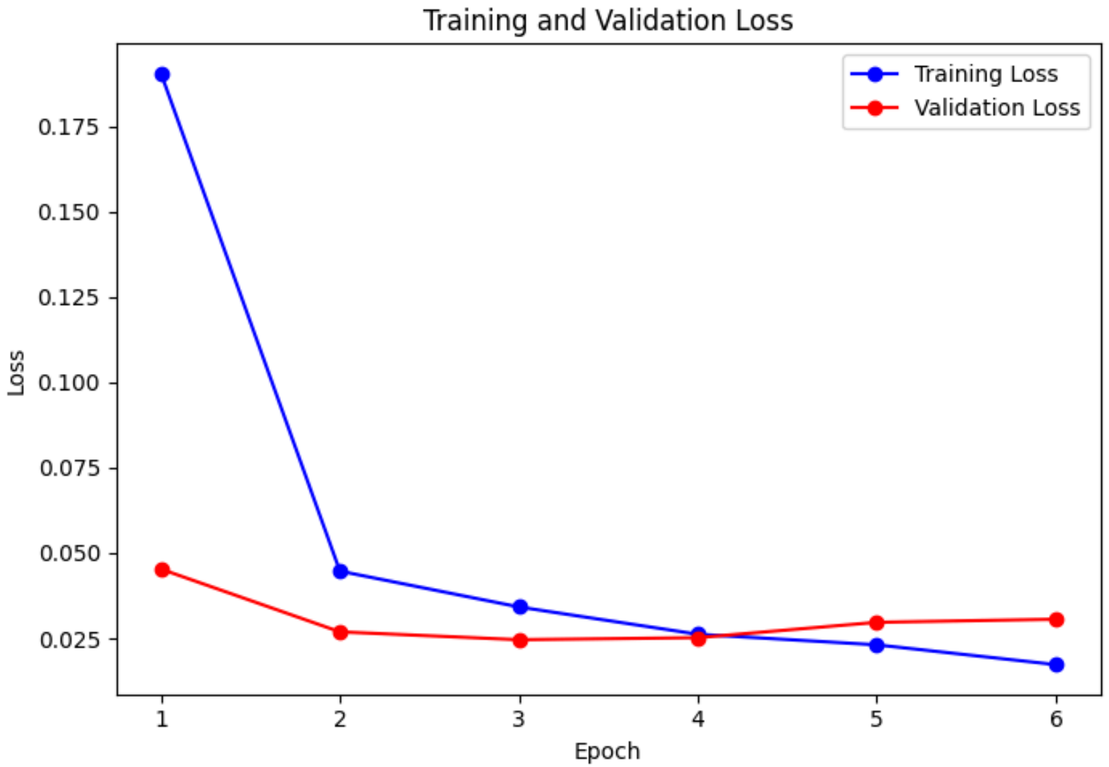
Batch Optimization and ReLu:

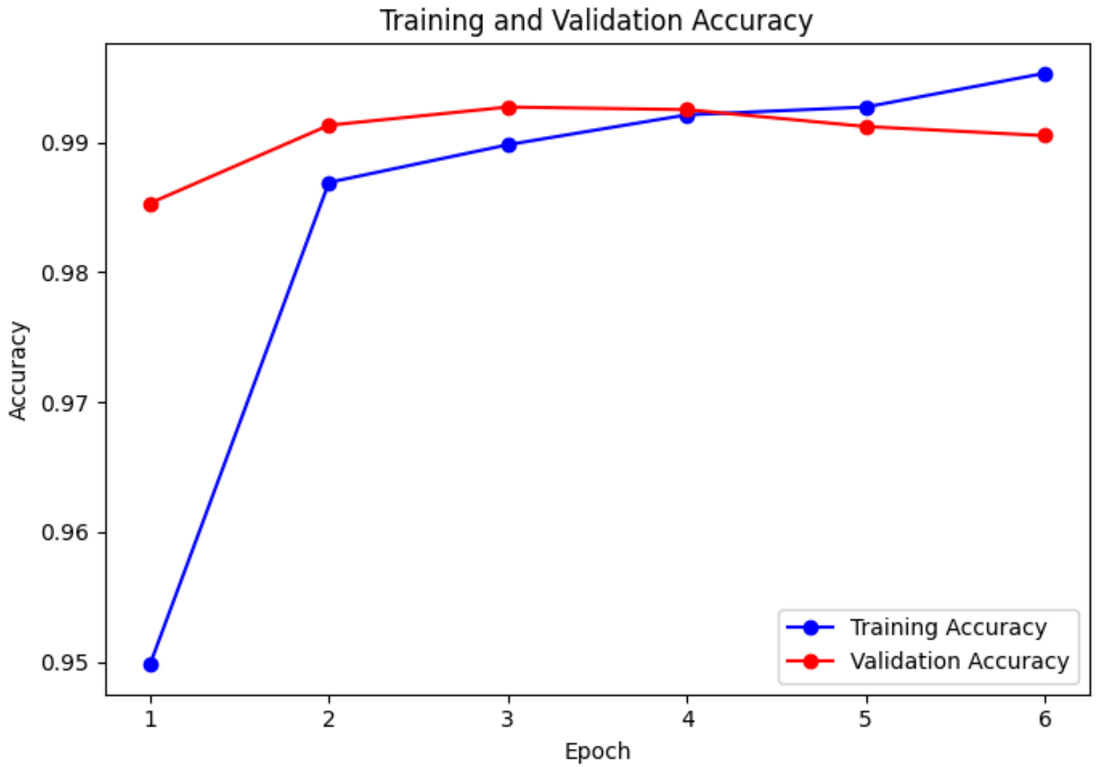
- Each convolutional layer is followed by batch normalization and ReLU activation for stable training.

Global Average Pooling (GAP):

- Replaces fully connected layers to reduce overfitting.

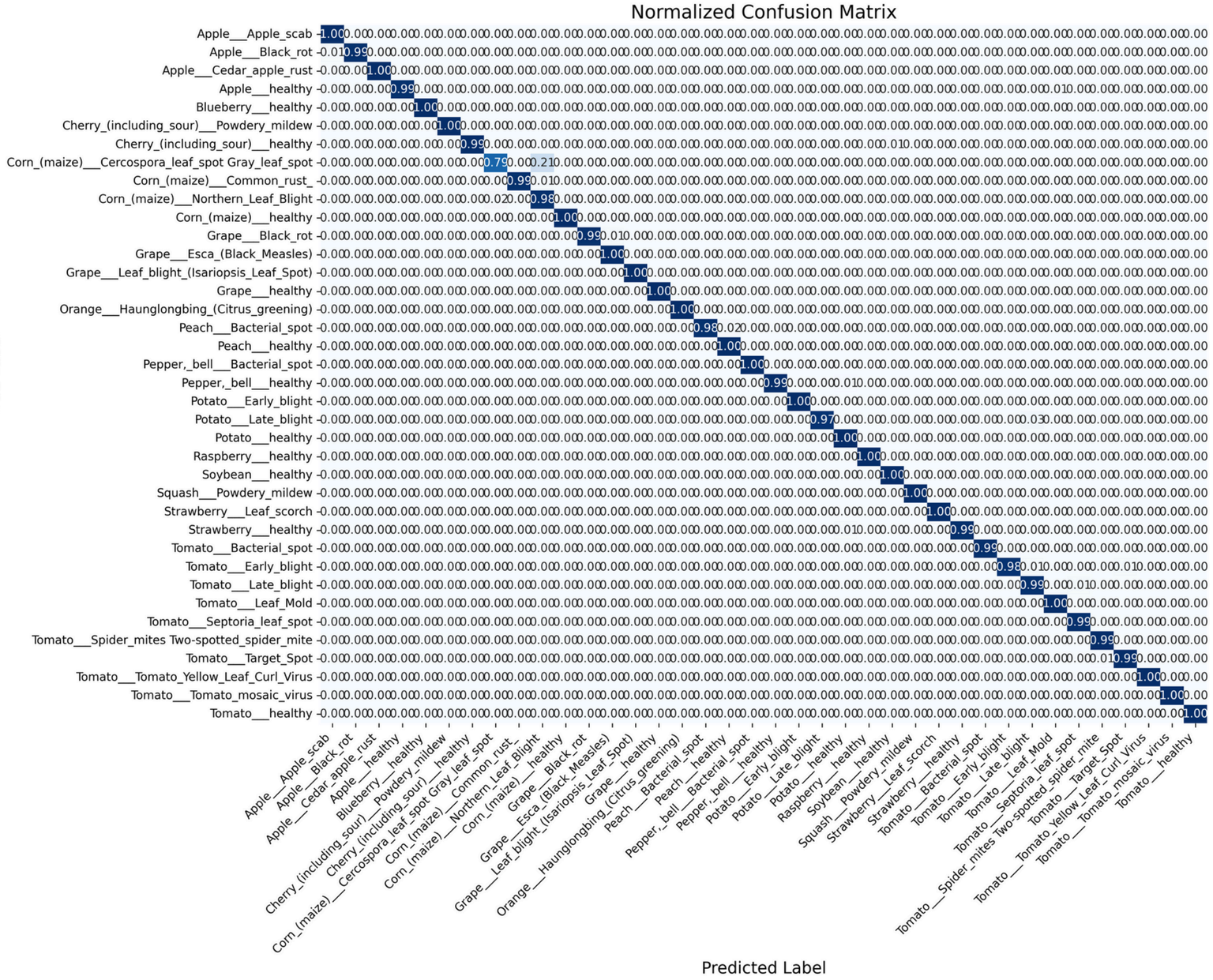
Training and Validation Accuracy and Loss





After the first epoch we see the most drastic variation in all the metrics, with a hefty stabilization after the second epoch for both for training and accuracy.

Confusion Matrix



This confusion matrix shows excellent classification performance:

- Diagonal dominance: Most values are concentrated along the diagonal, meaning the model correctly predicted nearly all classes.
- Normalization: Values range from 0 to 1, with 1.00 on the diagonal, indicating 100% accuracy for those classes.
- No significant off-diagonal values: Misclassifications are minimal or nonexistent.

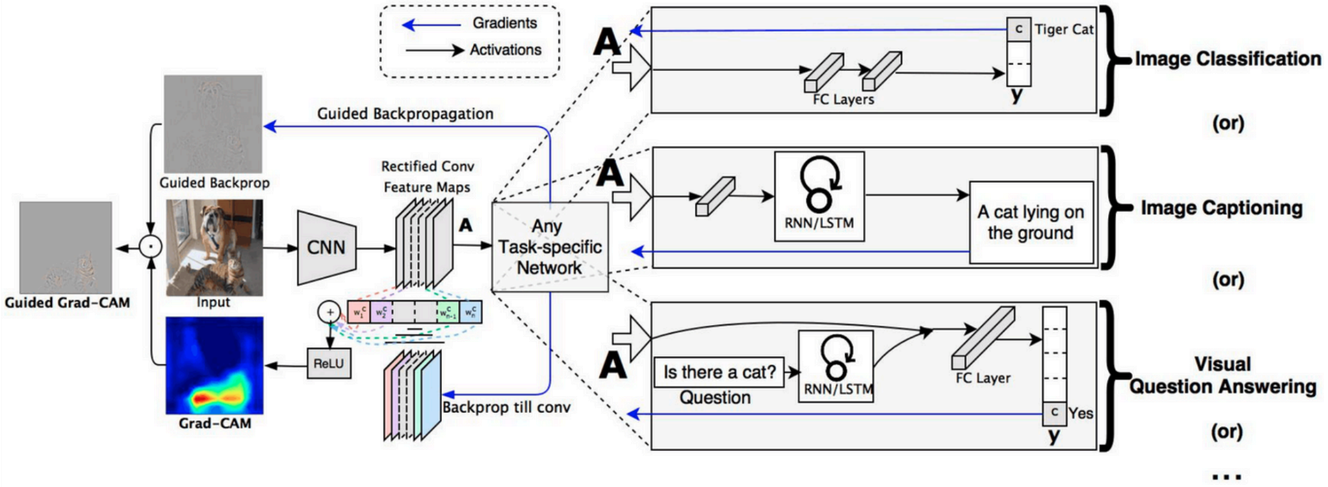
Summary: resnet50 shows strong generalization with a level of accuracy of 99%. The model displays perfect scores in both precision and recall for most classes; even classes with few sample (f.e. potato_healthy) are calssified with complete accuracy.

The Grad-Cam: Gradient-weighted Class Activation Mapping

In few words

Grad-CAM (Gradient-weighted Class Activation Mapping) allows us to visualize the regions of the imput image where the model focus the most for its predictions.
It works by analyzing the gradients of a target class score with respect to the feature maps in the last convolutional layer. These gradients show how important each feature map is for predicting that class.

This heatmap highlights the regions in the image that had the most influence on the model's decision: warmer colors (red/yellow) show where the model focused the most.



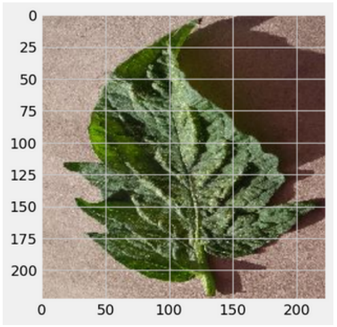
This process visually explains where the model is focusing its attention when analyzing an image—in this case, a leaf.

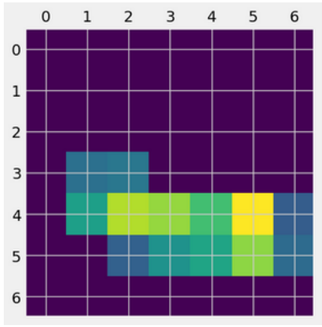
Here's what it does:

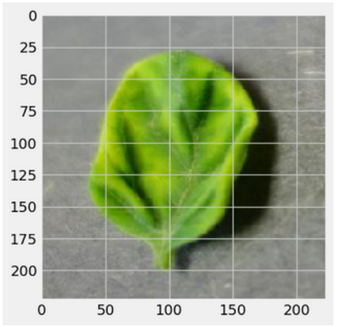
1. Generates a heatmap that highlights the most important areas in the image for the model's prediction.
2. Displays the heatmap on its own so you can see the intensity of focus across the image.
3. Overlays the heatmap on the original image to show exactly which parts of the leaf influenced the model's decision the most.

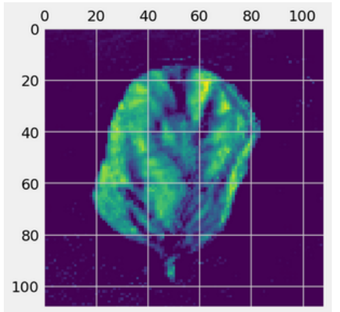
This is often used in deep learning to make models more interpretable.

Some examples









Thank you for the attention!!!