Monitoring Crop Health Using Computer Vision

Overview

Modern agriculture increasingly relies on drone imagery and artificial intelligence to monitor crop health efficiently and at scale. In this project, we will simulate the core computer vision component of an AI-driven drone system that detects unhealthy or stressed crop areas from aerial images.

We will use a publicly available simulated dataset of agricultural crops to build a Python program that detects **visibly unhealthy regions** using **image classification.**

Dataset used

New Plant Disease (link- https://www.kaggle.com/datasets/vipoooool/new-plant-diseases-dataset) dataset is recreated using offline augmentation from the original dataset. This dataset consists of about 87K RGB images of healthy and diseased crop leaves which is categorized into 38 different classes. The total dataset is divided into 80/20 ratio of training and validation set preserving the directory structure. A new directory containing 33 test images is created later for prediction purpose.

Pre-Processing Steps

The following preprocessing pipeline was applied to all images before training and evaluation:

Image Resizing:

- All input images resized to **(224 × 224)** pixels to match the model's expected input size.

Normalization:

- Pixel values scaled to the range **[0, 1]** by dividing by 255.0 to ensure stable gradient flow and faster convergence during training.

One-Hot Encoding:

- Labels converted to **categorical one-hot vectors**, suitable for multi-class classification using `categorical_crossentropy` loss.

Data Loading Pipeline:

- Used `tf.keras.preprocessing.image_dataset_from_directory()` to load images directly from folder structures organized by class.
- Applied `.map(preprocess)` to normalize images on-the-fly.
- Enabled `.prefetch(tf.data.AUTOTUNE)` for efficient data pipeline performance.

Note: Data Augmentation was not applied in this pipeline, since the dataset is already an Augmented version of real dataset.

Model Details

Two Models were used for the Classification of Images:

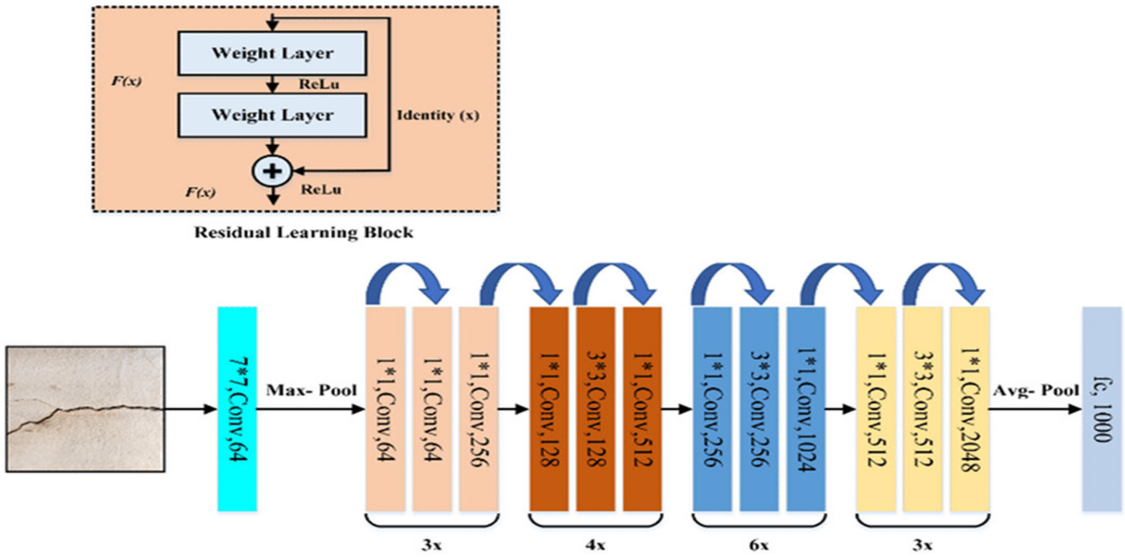
➢ ResNet50

ResNet-50 is a deep convolutional neural network built on the concept of **residual learning**. It uses **skip connections** (also called residual connections) to pass information across layers, helping prevent issues like vanishing gradients in deep networks. This allows ResNet-50 to effectively train models with **50+ layers**, making it powerful for complex image classification tasks. Developed by Kaiming He and team, ResNet revolutionized deep learning by enabling the training of ultra-deep models like ResNet-152.

• ResNet-50 Architecture

ResNet-50 consists of 50 layers that are divided into 5 blocks, each containing a set of residual blocks. The residual blocks allow for the

preservation of information from earlier layers, which helps the network to learn better representations of the input data.



- The following are the main components of ResNET:

1. Convolutional Layers

The first layer of the network is a convolutional layer that performs convolution on the input image. This is followed by a max-pooling layer that downsamples the output of the convolutional layer. The output of the max-pooling layer is then passed through a series of residual blocks.

2. Residual Blocks

Each residual block consists of two convolutional layers, each followed by a batch normalization layer and a rectified linear unit (ReLU) activation function. The output of the second convolutional layer is then added to the input of the residual block, which is then passed through another ReLU activation function. The output of the residual block is then passed on to the next block.

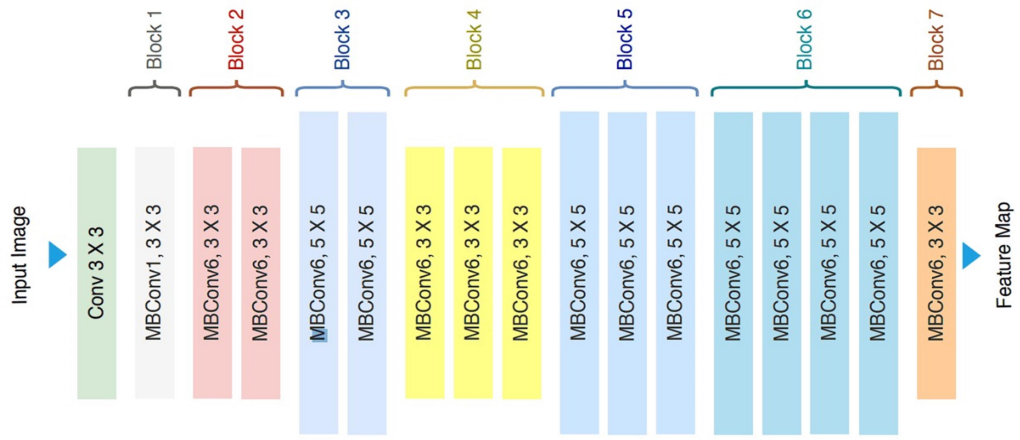
3. Fully Connected Layer

The final layer of the network is a fully connected layer that takes the output of the last residual block and maps it to the output classes. The number of neurons in the fully connected layer is equal to the number of output classes.

➢ EfficientNet B0:

- In the field of deep learning, the quest for more efficient neural network architectures has been ongoing. EfficientNet has emerged as a beacon of innovation, offering a holistic solution that balances model complexity with computational efficiency.
- EfficientNet-B0 is the baseline model of the EfficientNet family, proposed in the paper "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks" (arXiv:1905.11946). It was designed using Neural Architecture Search (NAS) to find an efficient and high-performing convolutional neural network.
- The model architecture is based on MBConv blocks (inverted residual blocks with depthwise separable convolutions), originally introduced in MobileNetV2. These blocks reduce computational cost while preserving performance. Additionally, Squeeze-and-Excitation (SE) blocks are used to adaptively recalibrate channel-wise feature responses, improving representational power.
- EfficientNet-B0 is notable for achieving a strong balance between accuracy and computational efficiency. With only 5.3 million parameters and ~390 million FLOPs, it achieves 77.1% top-1 accuracy on the ImageNet dataset—outperforming much larger models like **ResNet-50**.
- This baseline model serves as the foundation for the larger EfficientNet variants (B1 to B7), which are scaled versions using a compound scaling method to uniformly increase network depth, width, and resolution.

Architecture



- **Grad-Cam(**Gradient-Weighted Class Activation Map) **:**

The gradient-weighted class activation map (Grad CAM) produces a heat map that highlights important regions of an image using the target gradients (healthy, unhealthy) of the final convolutional layer.

How Grad-CAM Works:

Grad-CAM helps visualize **which parts of an image** a deep learning model focuses on while making a prediction. It works by:

1. **Passing the image through the model** to get predictions.
2. **Computing gradients** of the predicted class score with respect to the final convolutional layer.
3. **Weighting the feature maps** by these gradients to highlight important regions.
4. **Generating a heatmap** showing where the model is "looking" when predicting

Models setup :

- Model : ResNet50
- FrameWork : tensorflow 2.x
- Loss : Categorical_crossentropy
- Optimizer : Adam
- Epochs : 25(10 + 10 + 5)

- Model : EfficientNetB0
- FrameWork : tensorflow 2.x
- Loss : Categorical_crossentropy
- Optimizer : Adam
- Epochs : 75(30 + 20 +25)

Results and Interpretation

1. **Model Performance Overview:** Two deep learning models were trained:

Model	Validation Accuracy	Model Size	Notes
ResNet50	100%	Medium	Deeper network, faster convergence
EfficientNetB0	99.7%	Lightweight	Faster and more stable convergence

2. Classification Report:

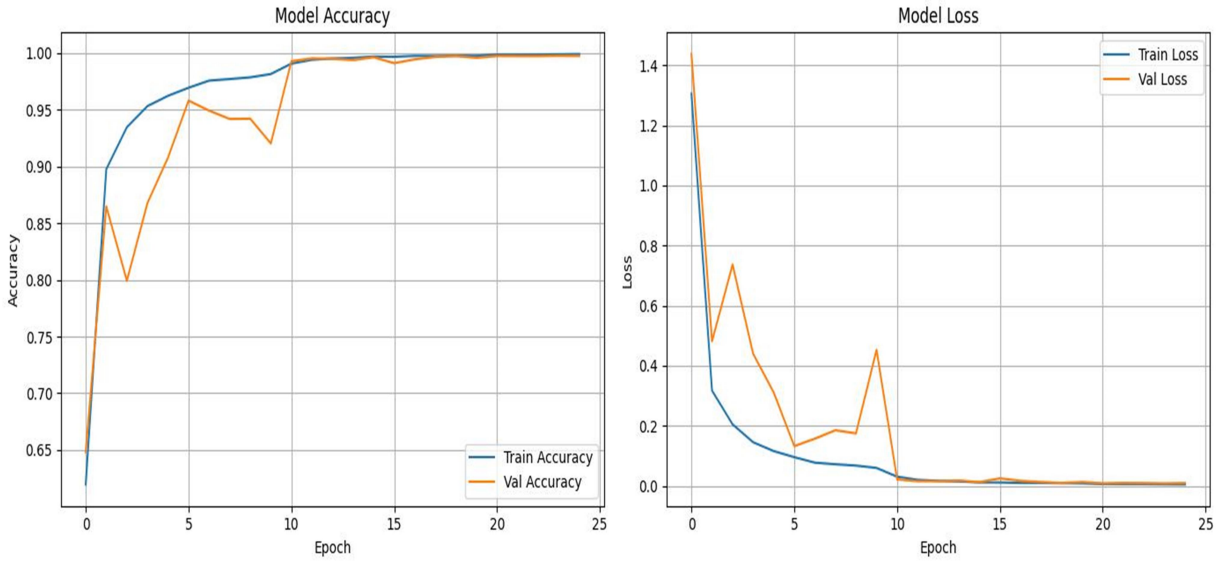
The table below summarizes the classification performance of the best-performing model (**ResNet50**) on the validation dataset.

	precision	recall	f1-score	support
Apple___Apple_scab	1.00	1.00	1.00	504
Apple___Black_rot	1.00	1.00	1.00	497
Apple___Cedar_apple_rust	1.00	1.00	1.00	440
Apple___healthy	1.00	1.00	1.00	502
Blueberry___healthy	1.00	1.00	1.00	454
Cherry_(including_sour)___Powdery_mildew	1.00	1.00	1.00	421
Cherry_(including_sour)___healthy	1.00	1.00	1.00	456
Corn_(maize)___Cercospora_leaf_spot Gray_leaf_spot	1.00	0.98	0.99	410
Corn_(maize)___Common_rust_	1.00	1.00	1.00	477
Corn_(maize)___Northern_Leaf_Blight	0.98	1.00	0.99	477
Corn_(maize)___healthy	1.00	1.00	1.00	465
Grape___Black_rot	1.00	1.00	1.00	472
Grape___Esca_(Black_Measles)	1.00	1.00	1.00	480
Grape___Leaf_blight_(Isariopsis_Leaf_Spot)	1.00	1.00	1.00	430
Grape___healthy	1.00	1.00	1.00	423
Orange___Haunglongbing_(Citrus_greening)	1.00	1.00	1.00	503
Peach___Bacterial_spot	1.00	1.00	1.00	459
Peach___healthy	1.00	1.00	1.00	432
Pepper,_bell___Bacterial_spot	1.00	1.00	1.00	478
Pepper,_bell___healthy	1.00	1.00	1.00	497
Potato___Early_blight	1.00	1.00	1.00	485
Potato___Late_blight	1.00	1.00	1.00	485
Potato___healthy	1.00	1.00	1.00	456
Raspberry___healthy	1.00	1.00	1.00	445
Soybean___healthy	1.00	1.00	1.00	505
Squash___Powdery_mildew	1.00	1.00	1.00	434
Strawberry___Leaf_scorch	1.00	1.00	1.00	444
Strawberry___healthy	1.00	1.00	1.00	456
Tomato___Bacterial_spot	1.00	1.00	1.00	425
Tomato___Early_blight	0.98	1.00	0.99	480
Tomato___Late_blight	1.00	0.99	0.99	463
Tomato___Leaf_Mold	1.00	1.00	1.00	470
Tomato___Septoria_leaf_spot	1.00	0.99	1.00	436
Tomato___Spider_mites Two-spotted_spider_mite	1.00	0.99	0.99	435
Tomato___Target_Spot	0.99	0.99	0.99	457
Tomato___Tomato_Yellow_Leaf_Curl_Virus	1.00	1.00	1.00	490
Tomato___Tomato_mosaic_virus	1.00	1.00	1.00	448
Tomato___healthy	1.00	1.00	1.00	481
accuracy			1.00	17572
macro avg	1.00	1.00	1.00	17572
weighted avg	1.00	1.00	1.00	17572

Interpretation:

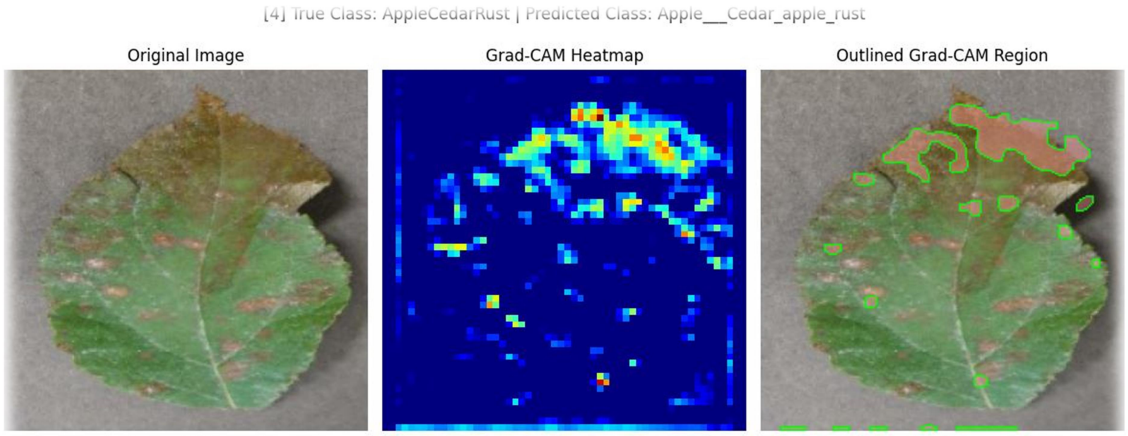
- The model shows balanced **precision** and **recall**, meaning it's good at both detecting unhealthy crops and avoiding false alarms.
- High **F1-score** suggests strong overall performance across classes.

3.Metrics Plot(ResNet50's):

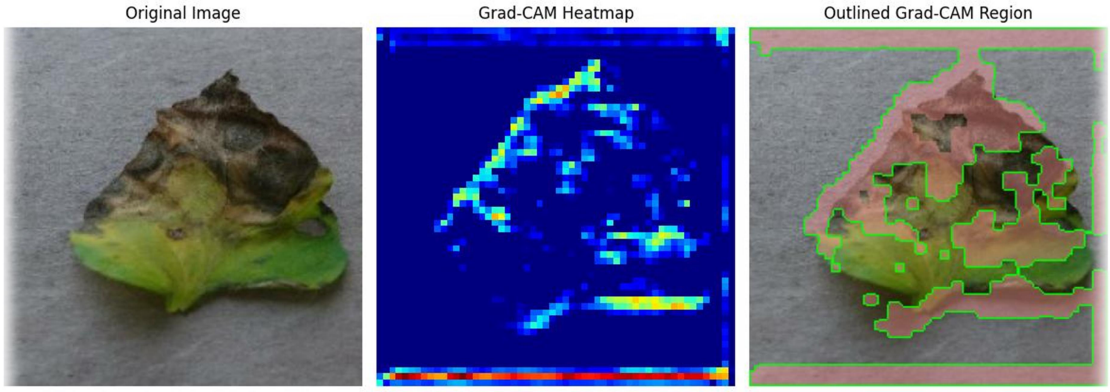


3. Visual Results Using Grad-CAM:

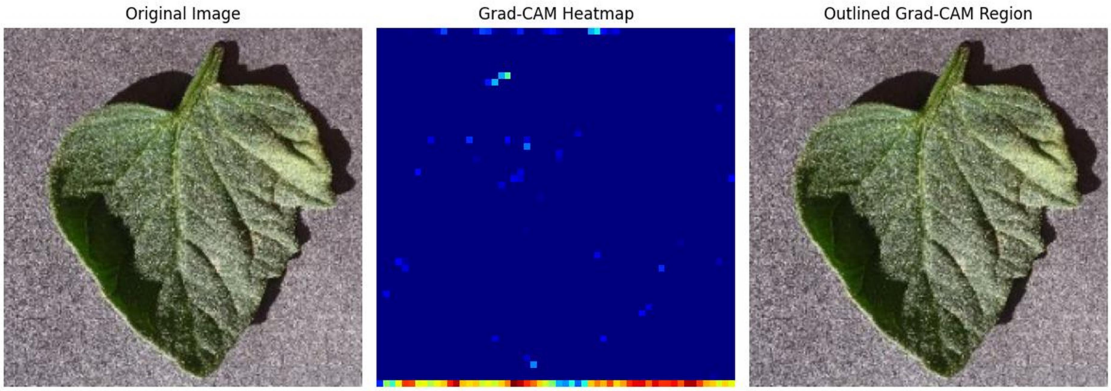
Grad-CAM visualizations help interpret the model's decision by highlighting regions that influenced the prediction.



[19] True Class: TomatoEarlyBlight | Predicted Class: Tomato__Early_blight



[24] True Class: TomatoHealthy | Predicted Class: Tomato__healthy



- **Red** areas show where the model is "looking"..

----------END OF REPORT----------