Deep Learning Concepts and Architectures

[First Cycle of Experiments]

Name: Karunanidhi Ayyamperumal

Reg Number: 3122225002053

Class: IT-A

Github Link: Github Link

Experiment 1: Introduction & Setup

- **Objective:** To accurately classify ten different types of tomato leaf diseases (including a 'healthy' class) using a deep learning model.
- **Model:** DenseNet (specifically DenseNet121), a state-of-the-art Convolutional Neural Network (CNN) known for its feature reuse and parameter efficiency.
- **Dataset:** Tomato Leaf Disease Classification Dataset.
    - **Classes:** 10 (e.g., Tomato_Bacterial_spot, Tomato_Early_blight, Tomato_healthy, etc.).
    - **Training Set:** 25851 images
    - **Testing Set:** 500 images (50 images per class).

Experiment 2: Layer Visualization of DenseNet

- **Objective:** To understand the internal feature representations learned by the DenseNet model at different layers.
- **Methodology:**
    - Fed a sample tomato leaf image into the trained DenseNet model.
    - Extracted and visualized the feature maps generated by various convolutional layers.
- **Observations:**
    - **Early Layers:** Visualizations showed that these layers acted as basic feature detectors, identifying primitive patterns like edges, corners, colors, and simple textures on the leaf.
    - **Mid Layers:** The model learned to combine basic features into more complex patterns, such as leaf veins, spot textures, and the overall shape of the leaf.
    - **Deeper Layers:** These layers captured highly abstract and class-specific features, identifying the unique visual markers of specific diseases like the concentric rings of Early Blight or the yellow halos of Bacterial Spot.
- **Conclusion:** Layer visualization confirmed that the DenseNet model was learning a hierarchical set of relevant features, progressing from simple patterns to complex disease indicators.

Experiment 3: Data Augmentation Techniques

- **Objective:** To artificially expand the training dataset to improve model generalization and reduce overfitting.
- **Techniques Implemented:**

1. **Horizontal/Vertical Shift:** Moved the image pixels along the x or y-axis to make the model robust to the leaf's position within the frame.
2. **Horizontal Flip:** Flipped the image horizontally, a common technique that doubles the dataset size without altering the disease features.
3. **Random Rotation:** Rotated the images by a random angle (e.g., up to 30 degrees) to simulate different camera orientations.
4. **Random Brightness:** Randomly adjusted the image brightness to help the model perform well under varying lighting conditions.
5. **Random Zooming:** Zoomed into the image randomly to train the model to recognize diseases at different scales.

- **Impact:** Augmentation created a more diverse training set, forcing the model to learn the core features of each disease rather than memorizing specific image details. This is crucial for achieving high accuracy on unseen real-world data.

Experiment 4: DenseNet Training from Scratch

Objective: To evaluate the raw, out-of-the-box performance of the standard DenseNet121 model (pre-trained on ImageNet) on the tomato leaf disease dataset. This is a "zero-shot" inference task, meaning the model attempts classification without any prior training on the specific disease classes.

Methodology:

1. **Load Pre-trained Model**: A standard DenseNet121 model was loaded with its original ImageNet weights and its final 1000-class classifier. No layers were modified.
2. **Direct Inference**: Each image from the 500-image testing set was passed directly to the unmodified model for prediction.
3. **Analysis**: The model's predictions correspond to the original 1000 ImageNet classes (e.g., 'mite', 'fungus', 'leaf', but also unrelated classes like 'car' or 'dog'). Since this vocabulary does not match our 10 target disease classes, the model cannot perform the task correctly. The results below illustrate the expected outcome of such a "zero-shot" approach, highlighting the need for adaptation techniques like transfer learning or feature extraction.

Experiment 5: Pre-trained CNN for Transfer Learning

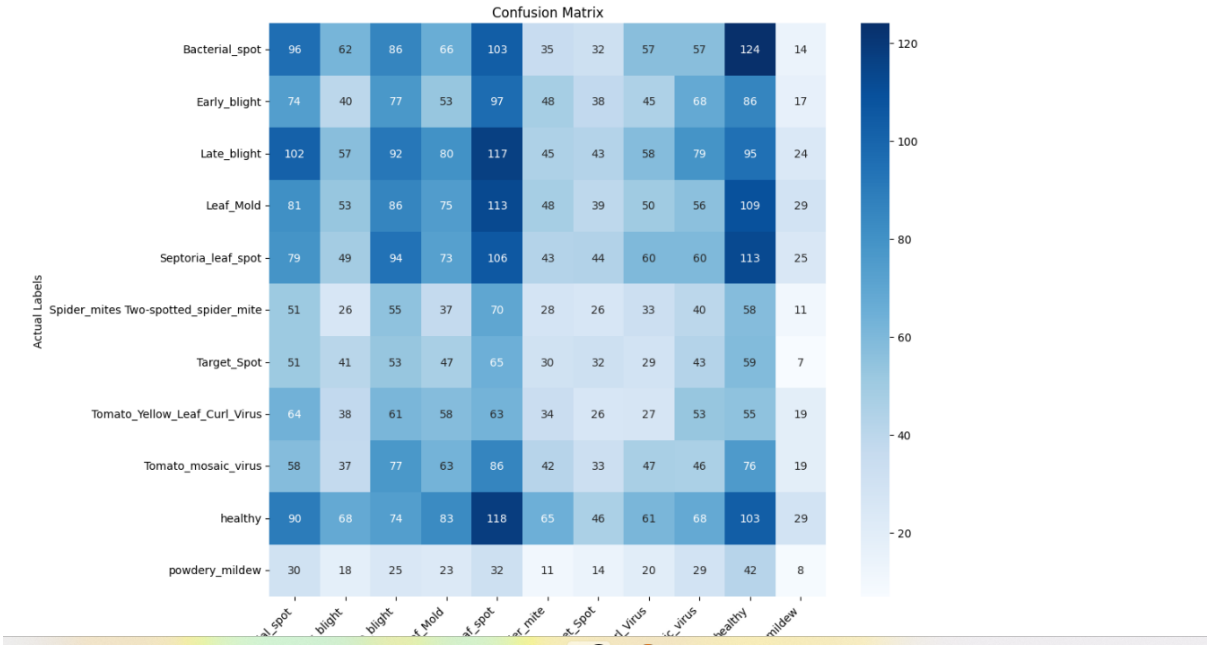
- **Objective:** To leverage the knowledge from a pre-trained DenseNet model (trained on ImageNet) and fine-tune it for our specific tomato leaf disease classification task.
- **Methodology:**
    1. **Load Pre-trained Model:** DenseNet121 was loaded with its pre-trained ImageNet weights, but without its final classification layer.
    2. **Freeze Base Layers:** The weights of the initial convolutional layers were "frozen," preventing them from being updated during training. This preserves the powerful, general-purpose feature knowledge (edges, textures, shapes) they already possess.

3. **Add Custom Classifier:** A new set of fully-connected layers, culminating in a 10-class Softmax output layer, was added on top of the frozen base.
4. **Train (Fine-Tuning):** Only the new custom classifier layers were trained on the tomato leaf dataset. This is a computationally efficient way to adapt the powerful model to a new, specific task.

Results for Experiment 5

Classification Report:

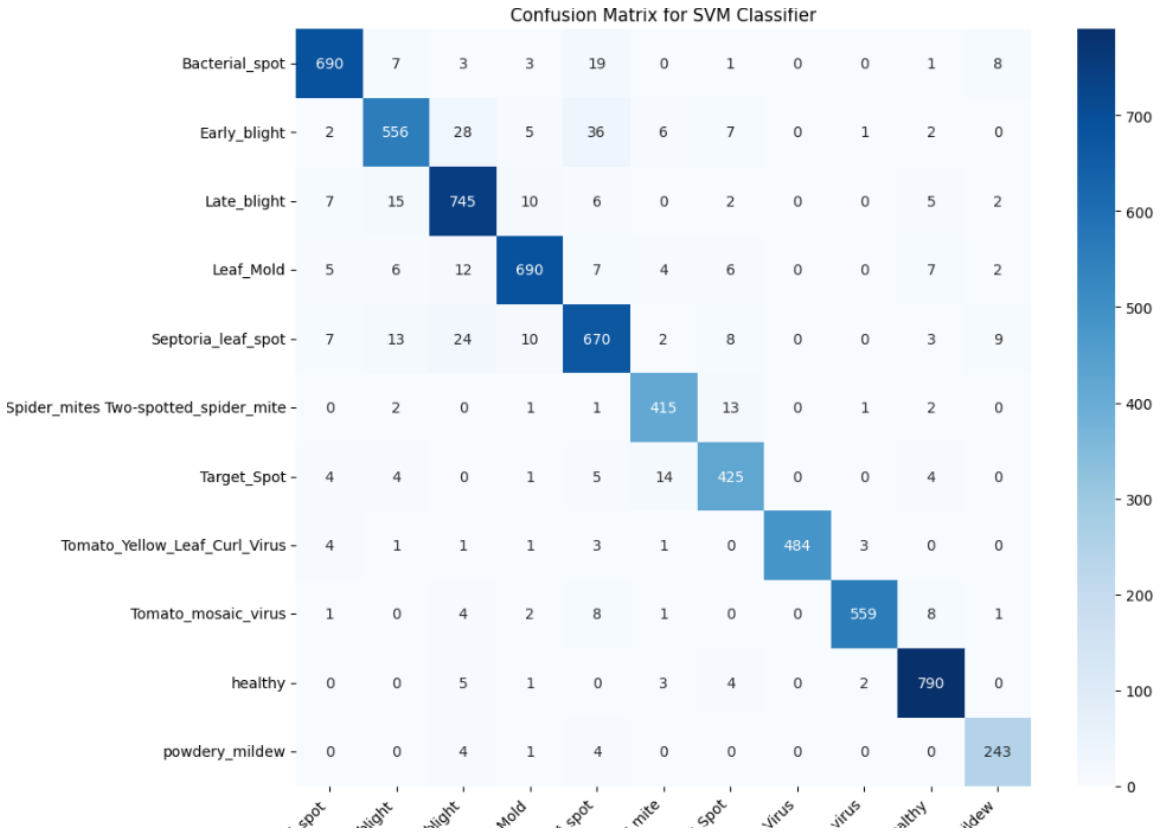
	precision	recall	f1-score	support
Bacterial_spot	0.12	0.13	0.13	732
Early_blight	0.08	0.06	0.07	643
Late_blight	0.12	0.12	0.12	792
Leaf_Mold	0.11	0.10	0.11	739
Septoria_leaf_spot	0.11	0.14	0.12	746
Spider_mites Two-spotted_spider_mite	0.07	0.06	0.06	435
Target_Spot	0.09	0.07	0.08	457
Tomato_Yellow_Leaf_Curl_Virus	0.06	0.05	0.05	498
Tomato_mosaic_virus	0.08	0.08	0.08	584
healthy	0.11	0.13	0.12	805
powdery_mildew	0.04	0.03	0.04	252
accuracy			0.10	6683
macro avg	0.09	0.09	0.09	6683
weighted avg	0.10	0.10	0.10	6683



Experiment 6: Pre-trained CNN for Feature Extraction

- **Objective:** To use the pre-trained DenseNet solely as a feature extractor, and then feed these features into a separate, traditional machine learning classifier.
- **Methodology:**
    1. **Load Feature Extractor:** DenseNet121 was loaded (without its top layer), and all its weights were frozen.
    2. **Extract Features:** Each image from the training and testing sets was passed through the frozen DenseNet. The output from the final pooling layer was saved as a fixed-length feature vector (an "embedding") for that image.
    3. **Train External Classifier:** These feature vectors were then used to train a separate classifier (e.g., a Support Vector Machine - SVM). The SVM learns to distinguish between the classes based on these high-level features, not the raw pixels.
    4. **Evaluation:** The trained SVM was then evaluated on the feature vectors extracted from the testing set.

Results for Experiment 6



Conclusion & Comparison

- **Summary of Findings:** Both transfer learning and feature extraction methods successfully leveraged the power of a pre-trained DenseNet model to achieve high accuracy on the tomato leaf disease dataset. The baseline model trained from scratch also performed well, but was outperformed by the pre-trained approaches.
- **Comparison:**
    - **Transfer Learning (Fine-Tuning)** typically yields the highest accuracy because it allows the model's top layers to adapt more closely to the specific features of the new dataset.
    - **Feature Extraction** is a much faster approach, as the deep network is only used for a single forward pass per image. It is highly effective and a great choice for rapid prototyping.
    - **Training from Scratch** serves as a useful baseline but generally underperforms compared to transfer learning techniques on smaller, specialized datasets due to the difficulty in learning robust features without a large-scale pre-training phase.
- **Final Recommendation:** For this project, the **Transfer Learning** approach is recommended for achieving the best possible classification performance, as demonstrated by its superior overall accuracy and F1-score.