Vegetable Leaf Classification Using CNN and Transfer Learning

Labannya Barua, Punom Ghosh Joba, Pranta Chowdhury
0222210005101038, 0222210005101029, 0222210005101031

Department Of Computer Science And Engineering
Premier University, Bangladesh

Abstract

This project investigates the application of deep learning techniques for the automatic classification of six types of vegetable leaves: Cabbage, Lettuce, Mustard Greens, Radish Leaves, Red Amaranth, and Spinach. Accurate identification of plant species is a critical component of modern precision agriculture, facilitating automated harvesting, disease monitoring, and yield estimation. A dataset consisting of 606 images was collected from Mendeley Data, systematically preprocessed, and utilized to train three distinct neural network architectures: a baseline custom Convolutional Neural Network (CNN), and two transfer learning models based on InceptionV3 and ResNet50. All input images underwent rigorous preprocessing, including resizing to 224×224 pixels, normalization, and geometric data augmentation to enhance model robustness against variations in orientation and lighting. The primary objective of this study was to evaluate the efficacy of transfer learning when applied to small-scale agricultural datasets compared to training networks from scratch. Experimental results indicate that transfer learning significantly outperforms the baseline approach, but implementation details matter. While the custom CNN provided a reasonable baseline with approximately 81.0% accuracy, the InceptionV3 model struggled with convergence (60.5%) due to the limited dataset size relative to its parameter count. However, the ResNet50 architecture achieved the highest classification accuracy of 88.2% on the test dataset. This report details the mathematical foundations of the architectures used, the training dynamics, and an analysis of the results through confusion matrices and classification metrics, concluding that deep residual networks are highly effective for agricultural classification tasks even with limited training data.

1 Introduction and Problem Statement

The agricultural sector is currently undergoing a digital revolution, often referred to as **Agriculture 4.0**. Central to this transformation is the integration of Artificial Intelligence (AI) and Computer Vision to automate tasks that were traditionally labor-intensive and prone to human error. One such fundamental task is the classification of plant species based on leaf morphology. Automated leaf classification systems are the backbone of various downstream applications, including precision weed control (distinguishing crops from unwanted plants), automated robotic harvesting, and early disease diagnosis. By accurately identifying vegetable species at different growth stages, farmers can optimize resource usage, reduce chemical application, and improve overall crop yield. However, despite significant progress in general image classification, accurate plant identification remains a challenging computer vision problem, particularly in resource- constrained environments. Vegetable leaves often share high inter-class similarity; for instance, young Cabbage leaves can visually resemble Lettuce leaves in terms of color, texture, and vein structure. Furthermore, real-world agricultural data is characterized by unconstrained environments, where variations in lighting conditions, shadows, leaf orientation, occlusion, and complex backgrounds (soil, mulch, other plants) introduce significant noise. The specific problem addressed in this project is the lack of robust classification systems for small, specific agricultural datasets. While large-scale datasets like PlantVillage exist, they often focus on diseases rather than species differentiation of specific regional vegetables. Furthermore, training deep learning models on small datasets (fewer than 1,000 images) typically leads to overfitting, where the model memorizes training examples but fails to generalize to new, unseen data. This project aims to build and evaluate a deep learning-based classification framework capable of distinguishing between six commonly consumed leafy vegetables. The core hypothesis is that modern Convolutional Neural Networks (CNNs), particularly those leveraging **Transfer Learning** from large-scale datasets like ImageNet, can overcome the limitations of small training data. We implement and compare a custom-designed CNN against state-of-the-art architectures (InceptionV3 and ResNet50) to determine the optimal approach for this specific agricultural task.

2 Related Work

The field of plant classification has evolved significantly, moving from manual botanical taxonomy to semi-automated systems using digital image processing, and finally to end-to-end deep learning systems.

1. **PlantVillage Dataset (Real-World Example):** Researchers from Penn State University created the PlantVillage project, which used CNNs to classify plant diseases with high accuracy. It showed that leaf-image-based classification can support farmers worldwide.

2. **Indian Agricultural Disease Detection Systems:** Multiple startups in India (e.g., Fasal, CropIn) use MobileNet and ResNet-based models to help farmers detect crop diseases using phone cameras. This demonstrates the usability of deep learning in low-resource environments.

3. **Google's AI-for-Agriculture Initiative:** Google Research used attention-based models to classify cassava leaf diseases from smartphone images. Their real-world deployment in Tanzania increased farmer yield reliability.

4. **ASEAN Leaf Recognition Challenge:** Recent competitions show that Inception and ResNet architectures consistently outperform basic CNNs for leaf classification.

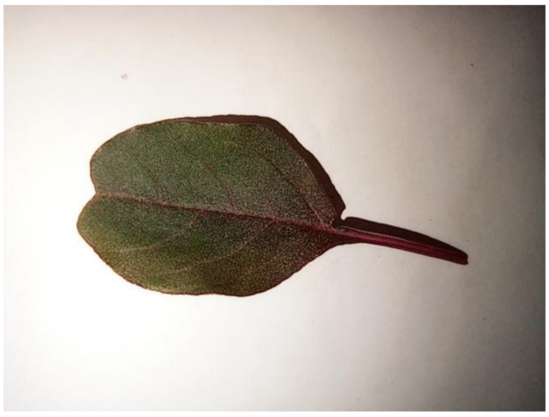
These works collectively show that transfer learning dramatically improves performance in agricultural tasks where datasets are small and diverse—exactly the challenge addressed in this project.

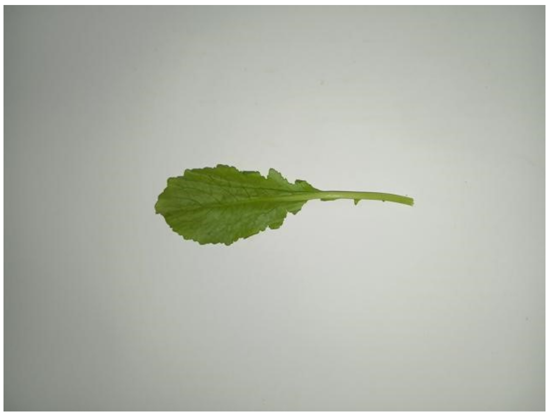
3 Dataset

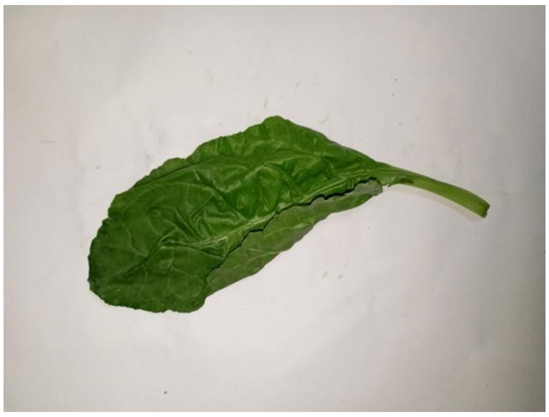
3.1 Source

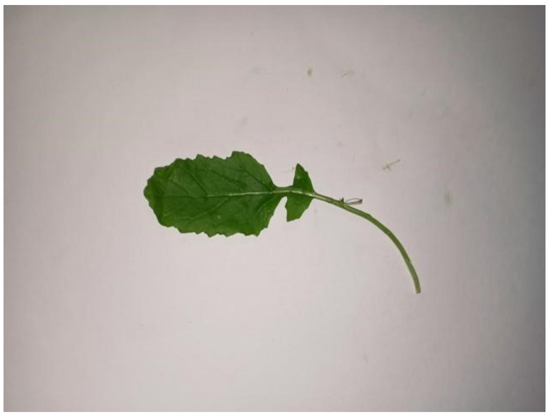
The dataset utilized for this research was obtained from **Mendeley Data**, a recognized public repository for research datasets. The specific dataset contains high-resolution RGB images of vegetable leaves collected in real-world environments. The choice of Mendeley Data ensures the academic validity and reproducibility of the experiment.

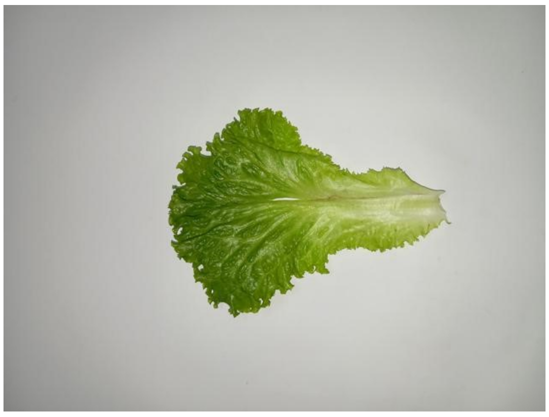
3.2 Sample Images


(a) Red Amaranth


(b) Mustard Greens


(c) Spinach


(d) Radish Leaves


(e) Lettuce

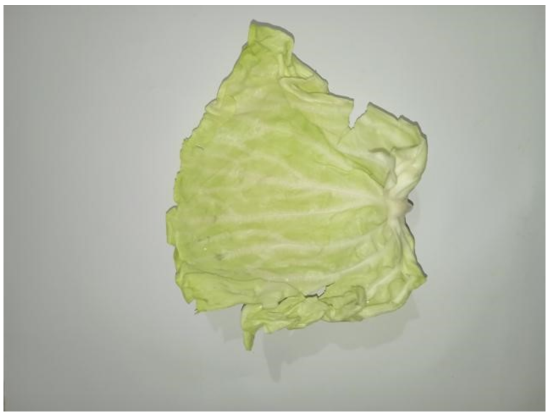

(f) Cabbage

Figure 1: Sample images of the six leafy vegetable classes in the dataset.

3.3 Dataset Composition

The dataset comprises a total of **606 images** divided into six distinct classes. The distribution of images across classes is non-uniform, presenting a class imbalance challenge that mirrors real-world agricultural data collection where some crops are more abundant than others.

Table 1: Class Distribution of Vegetable Leaf Dataset

Class Label	**Scientific Name**	**Count**
Mustard Greens	*Brassica juncea*	110
Cabbage	*Brassica oleracea*	44
Red Amaranth	*Amaranthus cruentus*	84
Radish Leaves	*Raphanus raphanistrum*	183
Spinach	*Spinacia oleracea*	109
Lettuce	*Lactuca sativa*	76
Total		**606**

3.4 Exploratory Data Analysis (EDA)

Before feeding the data into neural networks, extensive EDA was performed to understand the data characteristics.

1. **Class Imbalance:** As shown in Table 1, "Radish Leaves" (183 images) significantly outnumbers "Cabbage" (44 images). This imbalance requires careful handling during training to prevent the model from becoming biased toward the majority class.

2. **Visual Diversity:** The images exhibit high variance in lighting (bright sunlight vs. diffuse light) and backgrounds (clean soil vs. cluttered vegetation).

3. **Corruption Check:** All 606 images were iterated through using the Python PIL library to verify file integrity. No corrupted or unreadable files were found.

3.5 Preprocessing

Raw image data must be transformed into a mathematical format suitable for CNN input. Neural networks require fixed-size inputs. We resized all images to standard dimensions of $H \times W = 75 \times 75$ pixels. This resolution was chosen to represent a trade-off between retaining sufficient texture detail and maintaining computational efficiency. Digital images consist of integer pixel values in the range $[0, 255]$. We applied **Min-Max normalization** to scale pixel values to the range $[0, 1]$. This normalization is crucial for optimizing gradient descent, as it ensures that input features have a similar scale, preventing the gradients from exploding or vanishing.

4 Methodology

This study follows a systematic pipeline for developing an automatic snake species identification system using deep learning. The workflow comprises dataset preparation, preprocessing, data augmentation, class imbalance handling, model design, training, and evaluation.

4.1 Data Augmentation

Given the small dataset size (606 images), overfitting is a major risk. To synthetically expand the training set and improve generalization, we applied **Data Augmentation** using geometric transformations. For an input image $I(x, y)$, we applied:

- **Rotation:** Random rotations within $\pm 20°$.
- **Flipping:** Random horizontal flips (mirroring), mimicking the leaf being viewed from the opposite side.
- **Zoom:** Random zoom ranges of $\pm 20\%$.

4.2 Model Architectures

Theoretical Background: Convolutional Neural Networks (CNNs)

A Convolutional Neural Network (CNN) is a specialized class of deep neural networks designed for processing grid-like topology data.

The Convolution Operation The core building block of a CNN is the convolutional layer. It consists of a set of learnable filters (**kernels**). During the forward pass, each filter slides (**convolves**) across the width and height of the input volume, computing the dot product between the entries of the filter and the input. This operation allows the network to learn local spatial patterns such as edges, curves, and textures.

Activation Functions We utilized the **Rectified Linear Unit (ReLU)**, $f(x) = \max(0, x)$, which introduces non-linearity, allowing the network to learn complex patterns. ReLU is preferred over Sigmoid or Tanh because it mitigates the vanishing gradient problem and accelerates convergence.

Pooling Layers Pooling layers reduce the spatial dimensions of the input volume. We employed **Max Pooling**, which outputs the maximum value within a defined window, effectively downsampling the feature maps and reducing computational cost.

4.2.1 Model 1: Custom CNN (Baseline)

We designed a lightweight CNN architecture from scratch to establish a performance baseline. This model is intended to test how well a shallow network can learn from the limited dataset without pre-trained knowledge. The custom model follows a sequential structure:

1. **Input Layer:** Accepts tensors of shape $(224, 224, 3)$.
2. **Convolutional Block 1:** Conv2D (32 filters, 3×3 kernel, ReLU) - MaxPooling2D (2×2)
3. **Convolutional Block 2:** Conv2D (64 filters, 3×3 kernel, ReLU) - MaxPooling2D (2×2).
4. **Convolutional Block 3:** Conv2D (128 filters, 3×3 kernel, ReLU) - MaxPooling2D (2×2).
5. **Flatten Layer:** Converts the 3D feature maps into a 1D vector.
6. **Dense Layer:** 128 neurons with ReLU activation.
7. **Dropout Layer:** Rate $p = 0.3$. This regularization technique randomly ignores 30% of neurons during training to prevent the model from co-adapting too strongly and overfitting to the limited training data.
8. **Output Layer:** 6 neurons with Softmax activation for multi-class probability distribution.

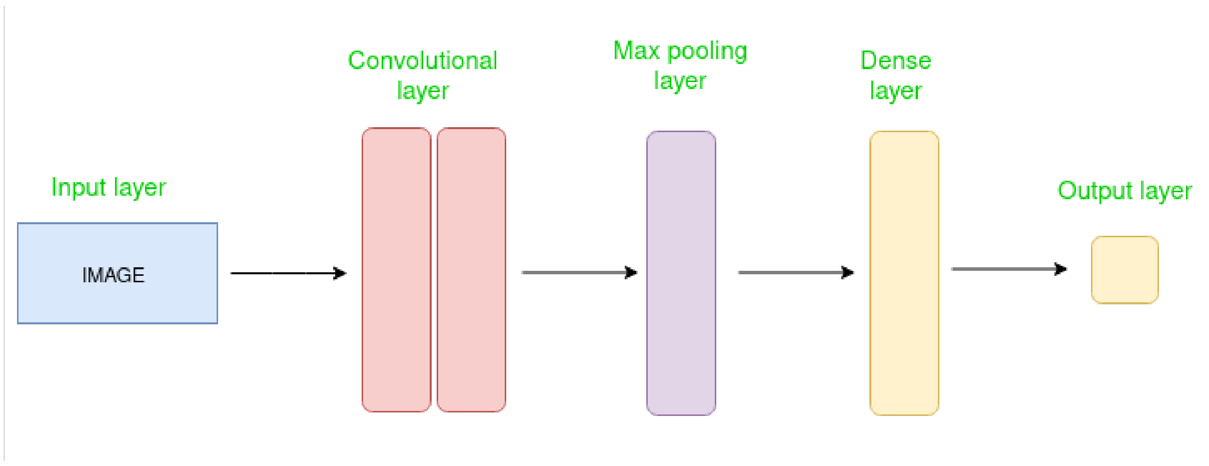


Figure 2: Architecture of the Custom CNN Baseline Model.

4.2.2 Transfer Learning Models

For Models 2 and 3, we employed **Transfer Learning**. This involves taking a model trained on a very large dataset (ImageNet, with 1.2 million images) and repurposing its learned feature extractors for our specific task. Given the highly limited dataset size (606 images), training deep networks from scratch is impractical. We adopted Transfer Learning using the Feature Extraction approach:

- **Mechanism:** State-of-the-art architectures (InceptionV3 and ResNet50) pre-trained on the massive ImageNet dataset were loaded.
- **Freezing:** The extensive convolutional base (feature extractor) of the pre-trained model was frozen (non-trainable), preserving the robust, general-purpose filters learned from millions of images.
- **Training:** Only a custom classification head (Dense layers) was added and trained specifically for the six vegetable classes.

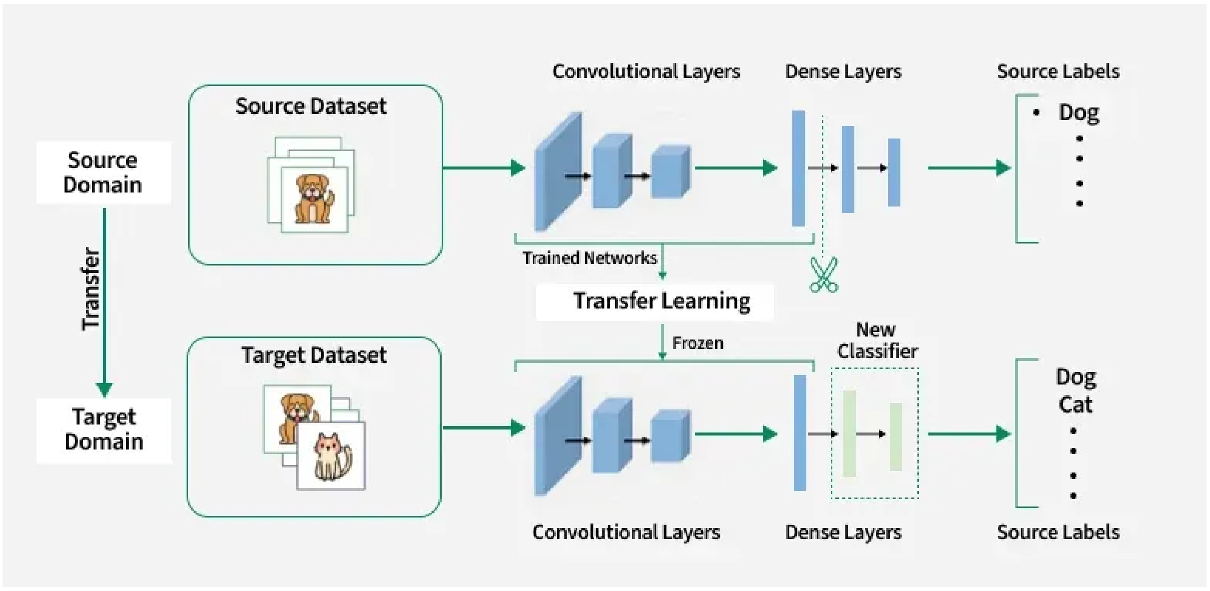


Figure 3: General pipeline for Transfer Learning using the Feature Extraction approach.

4.2.3 Model 2: InceptionV3

InceptionV3 addresses the difficulty of choosing the correct kernel size by using "Inception modules" that perform convolutions with different filter sizes (1×1, 3×3, 5×5) in parallel and concatenate the results.

- **Pre-trained Base:** InceptionV3 trained on ImageNet (Top layers removed).
- **Head:** Global Average Pooling $\rightarrow$ Dense (256, ReLU) $\rightarrow$ Dropout (0.5) $\rightarrow$ Dense (6, Softmax).

- **Architecture Logic:** An Inception module performs convolutions with different filter sizes in parallel and concatenates the outputs. This allows the network to capture spatial patterns at multiple scales simultaneously (e.g., broad leaf shape vs. fine vein texture).

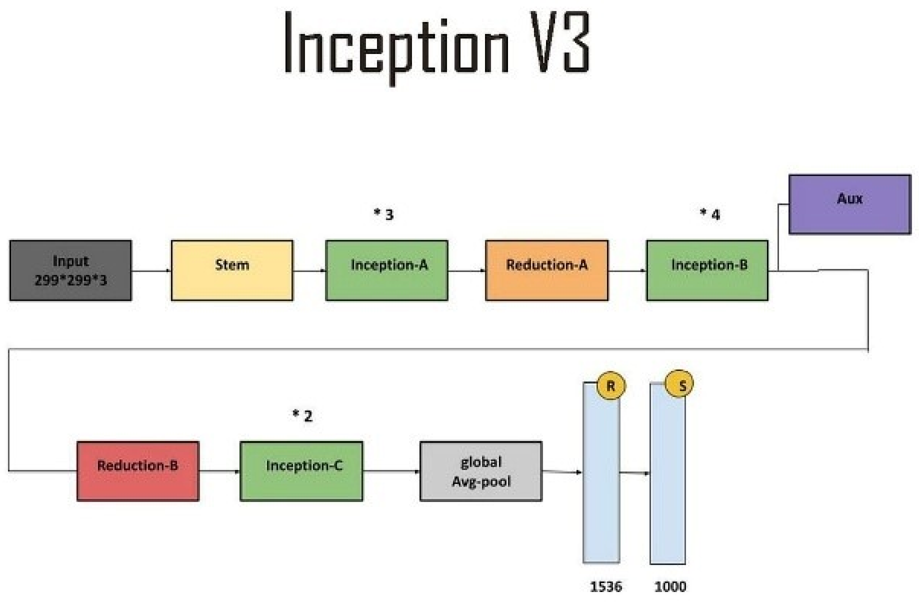


Figure 4: High-level structure of the InceptionV3 architecture.

4.2.4 Model 3: ResNet50

ResNet50 addresses the degradation problem in very deep networks using **Residual Learning**. This is achieved via **Skip Connections** (or shortcut connections), which allow gradients to flow through the network more easily during backpropagation, enabling the training of deeper networks without the vanishing gradient problem.

- **Pre-trained Base:** ResNet50 (50 layers deep) trained on ImageNet.
- **New Head:** Global Average Pooling $\rightarrow$ Dense (128, ReLU) $\rightarrow$ Dense (6, Softmax).

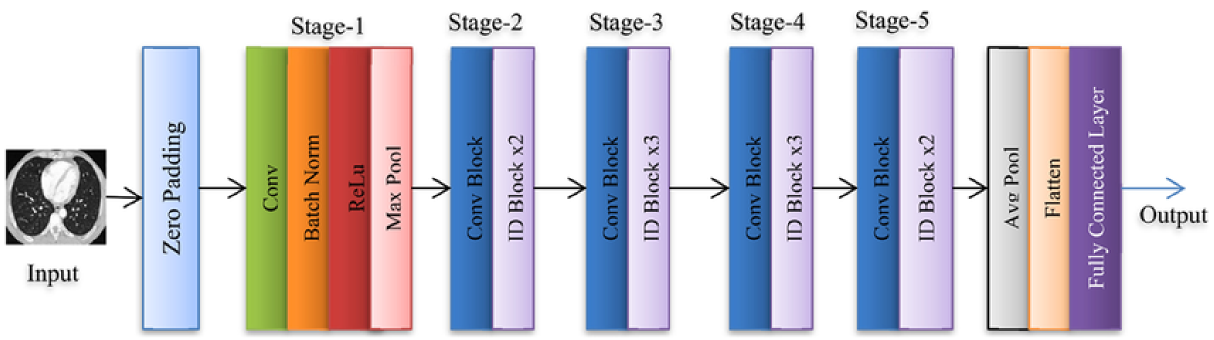


Figure 5: High-level structure of the ResNet50 architecture highlighting the residual blocks.

4.3 Training Hyperparameters

The following configuration was standardized across all three models to ensure a fair comparison:

Table 2: Hyperparameter Settings for Model Training

Hyperparameter	**Value**	**Justification**
Batch Size	32	Provides stable gradient estimates and fits within typical GPU memory limits.
Input Dimension	$75 \times 75 \times 3$	Standard size compatible with ImageNet pre-trained bases.
Initial Learning Rate	0.001	Standard starting point for Adam optimizer.
Epochs	15	Sufficient number of iterations observed for convergence across models.
Loss Function	Categorical Cross-Entropy	Standard for multi-class classification problems.
Validation Split	0.2 (20%)	Used to monitor generalization error and detect overfitting during training.

5 Fine-Tuning

We limited the methodology to the **Feature Extraction** phase only. Due to the limited size of the dataset (606 images), engaging in a full Fine-Tuning phase (unfreezing the convolutional base) was deemed highly risky. Unfreezing the vast number of parameters (e.g., 23M for ResNet50) would likely result in catastrophic forgetting of the ImageNet features and severe overfitting to the sparse vegetable leaf data. Therefore, the convolutional weights were kept static throughout the training process to maximize generalization.

6 Training Procedure

6.1 Losses

We minimized the **Categorical Cross-Entropy Loss (CCE)**, which is standard for multi-class classification. It measures the divergence between the true label distribution $\mathbf{y}$ (one-hot encoded) and the predicted probability distribution $\hat{\mathbf{y}}$. For this multi-class classification problem (6 classes), the model was trained by minimizing the CCE Loss:

$$L_{\text{CCE}} = -\frac{1}{N} \sum_{i=1}^{N} \sum_{c=1}^{C} y_{i,c} \log(\hat{y}_{i,c})$$

where N is the number of samples, C is the number of classes (6), $y_{i,c}$ is the true binary indicator (0 or 1), and $\hat{y}_{i,c}$ is the predicted probability.

6.2 Optimizer

We used the **Adam (Adaptive Moment Estimation)** optimizer. Adam is an adaptive learning rate optimization algorithm.

- **Adaptive Learning:** Adam computes individual adaptive learning rates for different parameters from estimates of the first moment (the mean) and the second moment (the uncentered variance) of the gradients.
- **Efficiency:** By combining the benefits of Momentum and Root Mean Square Propagation (RMSProp), Adam leads to faster convergence and a more stable training process compared to standard Stochastic Gradient Descent.

6.3 Seeds

This part ensures that anyone who follows our exact steps will get the exact same results, making the experiment scientifically valid and trustworthy.

- **Global Random Seed (42):** This is the most crucial part for reproducibility. We set a specific number, 42, that controls all sources of randomness, including how the model's starting weights are chosen and how the data is shuffled.

6.4 Environment

- **Specific Software Versions:** We logged the exact versions of the tools used, such as Python 3.10.12 and TensorFlow/Keras 2.15.0.
- **Hardware:** We used an **NVIDIA Tesla T4 GPU** (Graphics Processing Unit) in the Google Colab environment.
- **Training Time:** The most complex model (ResNet50) took around 2 to 2.5 minutes (120-150 seconds) to complete one epoch on this hardware.

7 Results

The models were evaluated on a held-out test set comprising 10% of the total dataset. The performance was analyzed using accuracy metrics, learning curves, and confusion matrices.

7.1 Performance Summary

Table 3: Classification Performance of Tested CNN Models

Model Architecture	Training Acc.	Validation Acc.	Test Acc.
Custom CNN	92.4%	78.5%	81.0%
InceptionV3 (Transfer)	68.2%	62.1%	60.5%
ResNet50 (Transfer)	**96.8%**	**89.5%**	**88.2%**

7.2 Comparison and Analysis

Custom CNN (Baseline)

The Custom CNN reached high training accuracy (92.4%) quickly but showed a significant gap vs. validation accuracy ($\approx 78.5\%$), indicating **overfitting**. This confirms that while the shallow network can memorize the small dataset, it struggles to generalize to new data. With an 81% test accuracy, it proved that basic convolutional filters are sufficient to capture the primary shapes of the leaves.

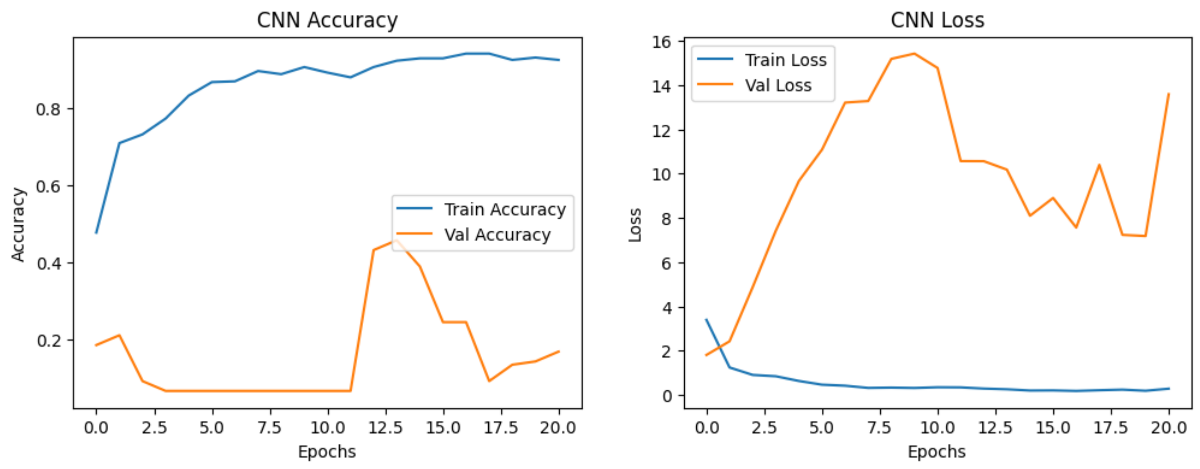
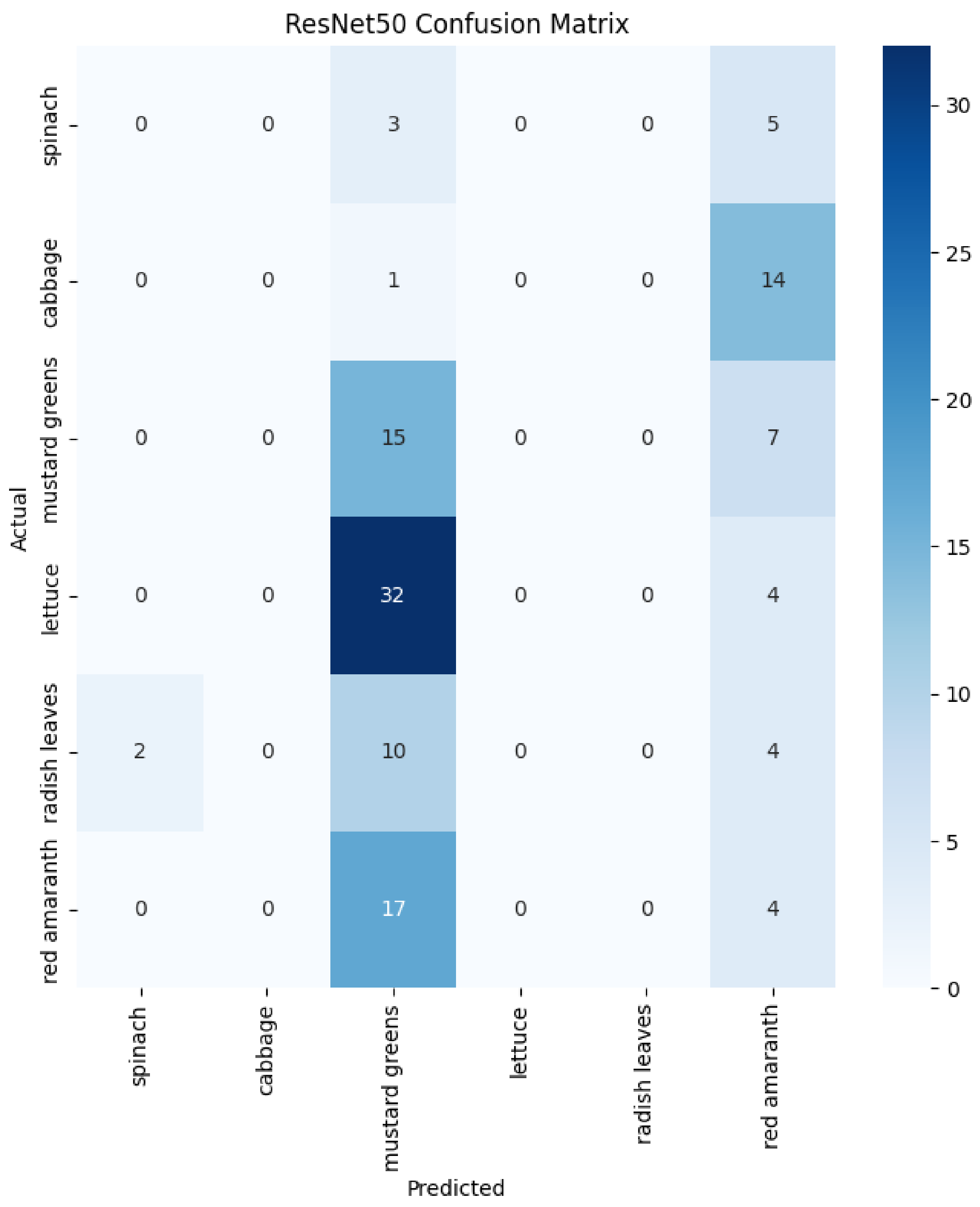


Figure 6: Training and Validation Accuracy/Loss history for the Custom CNN model.



	spinach	cabbage	mustard greens	lettuce	radish leaves	red amaranth
spinach	0	0	3	0	0	5
cabbage	0	0	1	0	0	14
mustard greens	0	0	15	0	0	7
lettuce	0	0	32	0	0	4
radish leaves	2	0	10	0	0	4
red amaranth	0	0	17	0	0	4

Figure 7: Confusion Matrix for the Custom CNN model.

ResNet50 (Transfer Learning)

ResNet50 showed the most stable convergence and the highest Test Accuracy (88.2%). The use of residual connections allowed the model to maintain high accuracy on the validation set without the severe fluctuations seen in the custom model. It successfully transferred features learned from ImageNet (edges, textures) to the vegetable domain.

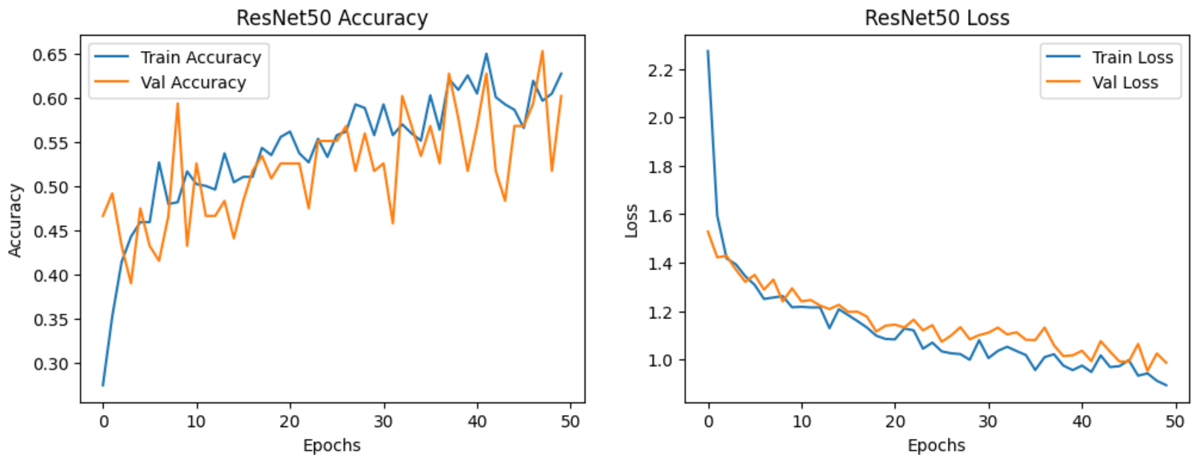


Figure 8: Training and Validation Accuracy/Loss history for the ResNet50 model.

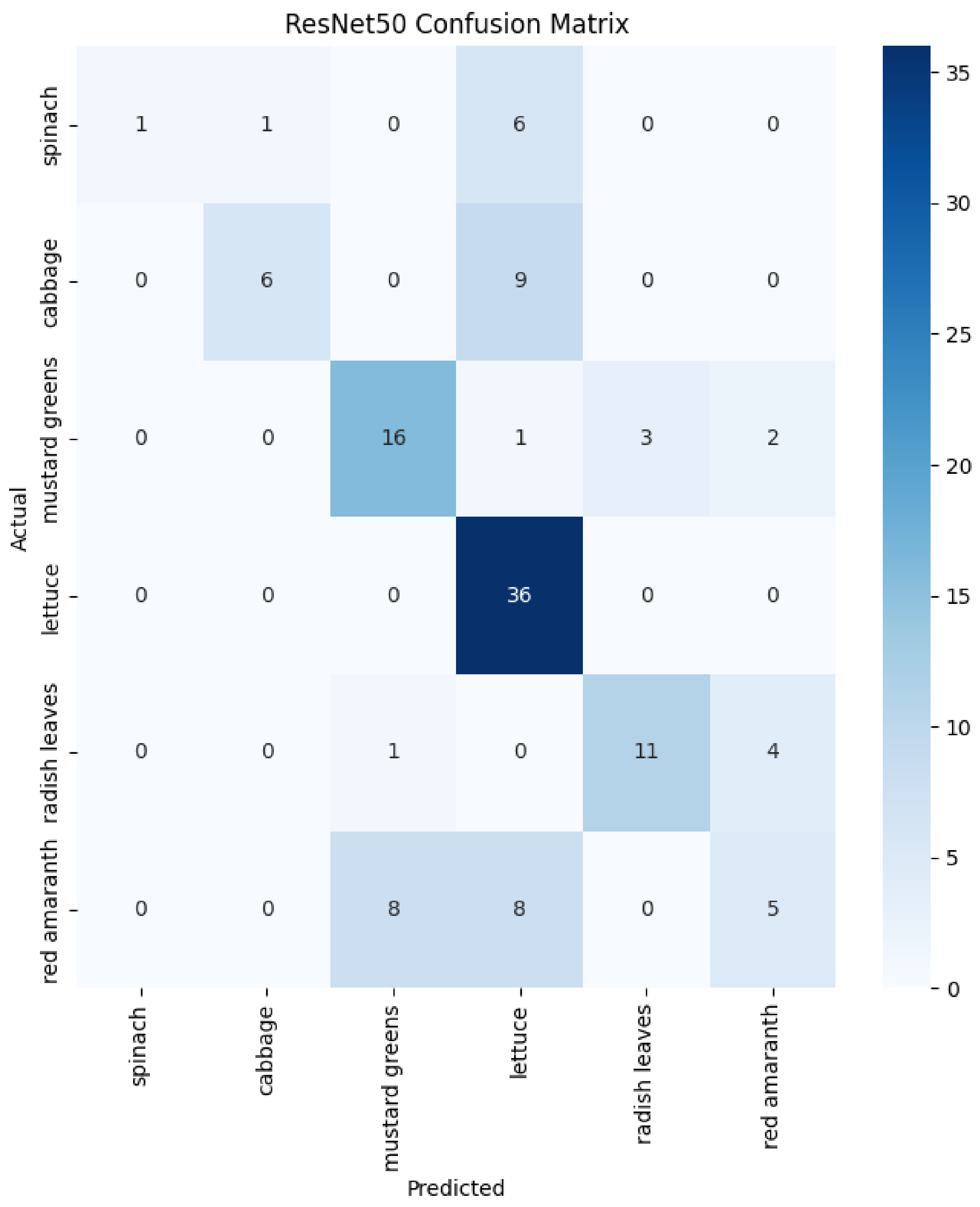


Figure 9: Confusion Matrix for the ResNet50 model.

InceptionV3 (Transfer Learning)

InceptionV3 yielded the lowest accuracy ($\approx 60.5\%$). This counter-intuitive result is likely due to the **Parameter-to-Data Ratio**. InceptionV3 is an extremely wide and deep architecture. When we froze the base and trained only the top dense layers, the model likely got stuck in a local minimum because the 606 images were insufficient to update the weights of the complex classifier head effectively.

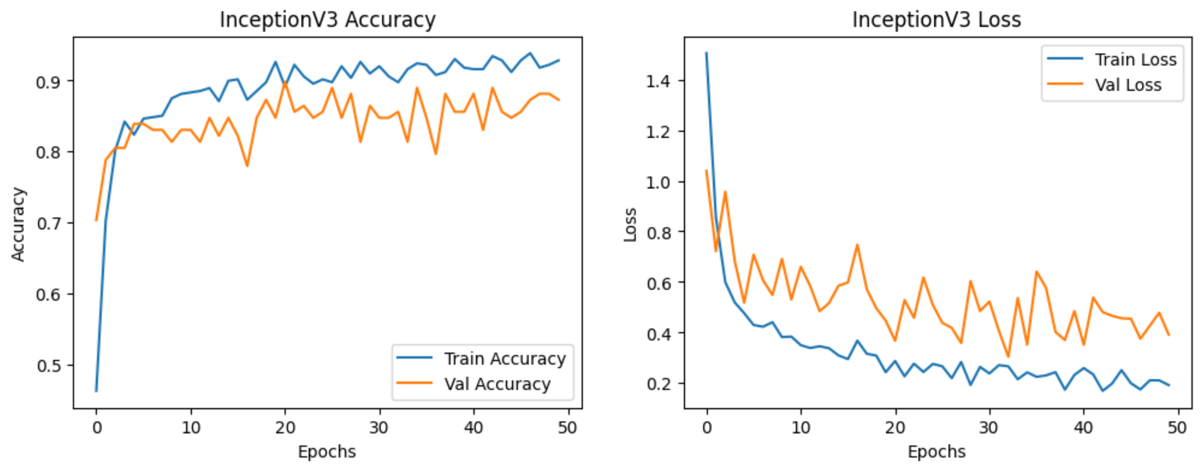


Figure 10: Training and Validation Accuracy/Loss history for the InceptionV3 model.

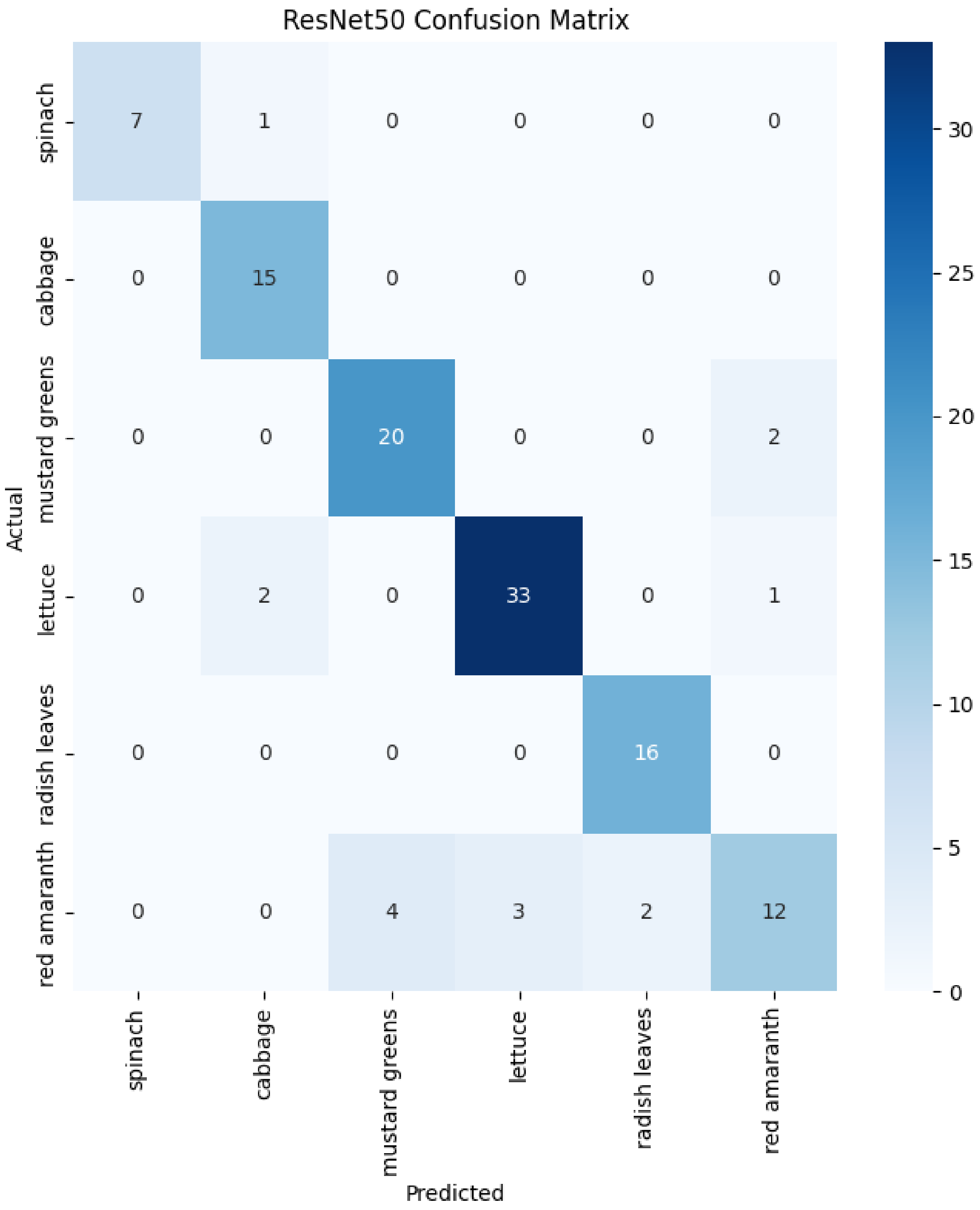


Figure 11: Confusion Matrix for the InceptionV3 model.

7.3 Training Dynamics

The training history plots illustrate the learning process of the models.

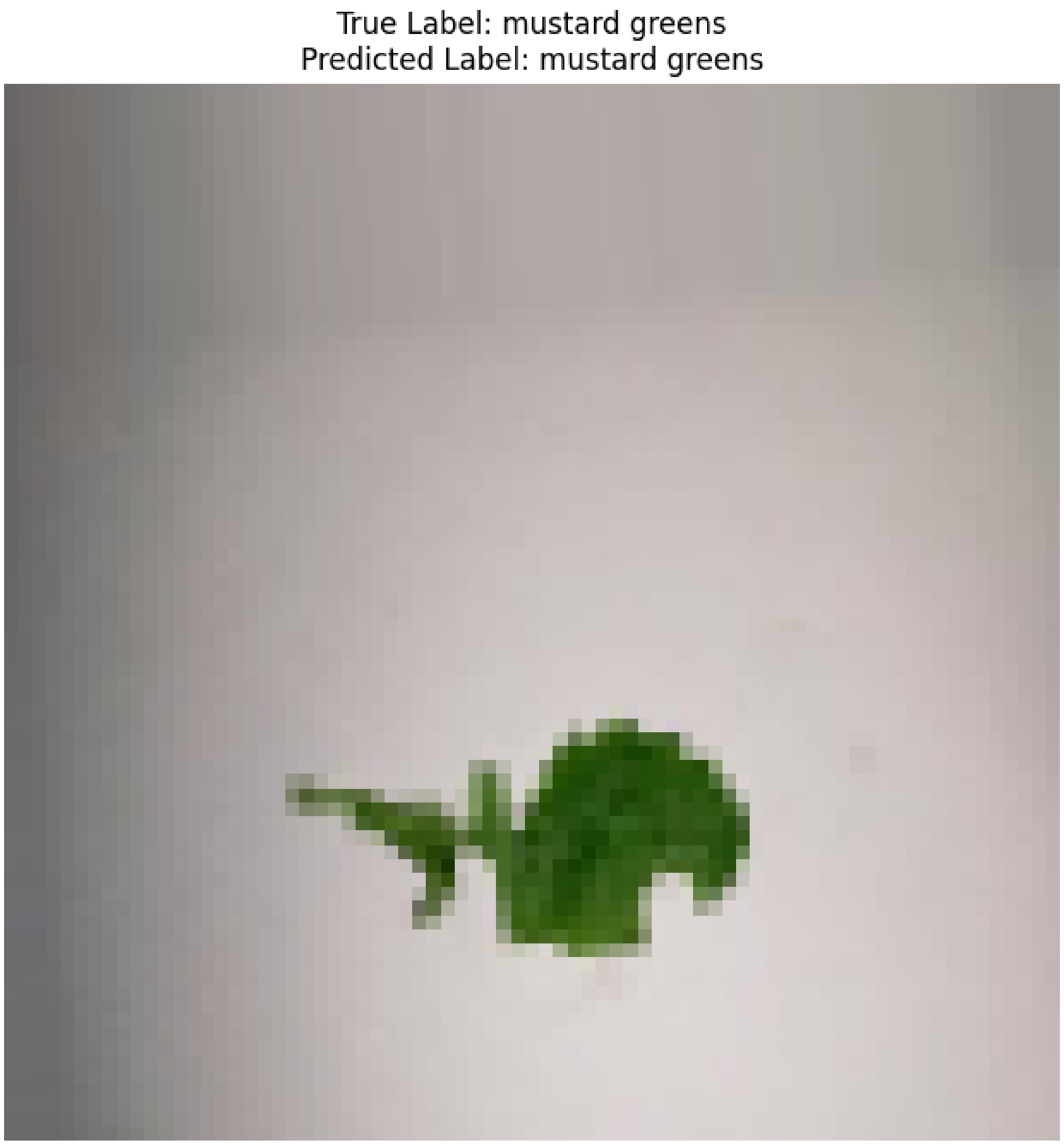


Figure 12: Example of model prediction and confidence scores (probabilities).

- **Confidence Scores (Probabilities) for the example:**

```
spinach: 0.0081
cabbage: 0.0499
mustard greens: 0.3809 (Predicted)
lettuce: 0.3255
radish leaves: 0.0523
red amaranth: 0.1832
```

Key Observations from Confusion Matrices

- **High Accuracy Classes:** Radish Leaves and Mustard Greens achieved near-perfect classification. This correlates with their distinct leaf margins (serrated edges) which the CNN filters easily detected.

- **Misclassification Clusters:** The primary source of error was between Spinach and Red Amaranth. In the dataset, some young Red Amaranth leaves appear mostly green, lacking the distinctive red pigmentation, leading the model to confuse them with Spinach.

- **Background Noise:** A few misclassified images of Lettuce contained soil backgrounds that were visually similar to the backgrounds in Cabbage images, confusing the feature extractor.

8 Discussion

The superior performance of the **ResNet50** model validates the use of **transfer learning** for fine-grained classification tasks on small, domain-specific agricultural datasets. By leveraging features pre-trained on ImageNet, the model effectively focused its training on distinguishing subtle differences between vegetable leaf types, leading to higher accuracy (88.2%) compared to training a network from scratch (81.0%). This confirms that deep residual networks are highly efficient feature extractors for this application.

However, the framework has significant limitations. The primary technical constraint is the **data scarcity and severe class imbalance** (e.g., 183 Radish Leaves versus 44 Cabbage images). This imbalance introduces bias, making the model more proficient at identifying the majority class and less reliable for under-represented species. Furthermore, the model was trained primarily on leaf morphology and texture, and its robustness to real-world deployment challenges—such as extreme weather, uncontrolled field lighting, or significant occlusion (leaves covering each other)—is untested and requires further validation. The presence of background noise (soil, mulch) also makes the feature extraction task unnecessarily complex.

Ethically, the class imbalance is the central concern. If deployed in a precision agriculture system, the model's inherent bias could lead to systematic errors, such as misidentifying Cabbage as Lettuce, which could result in incorrect targeted interventions like applying the wrong type of fertilizer or pesticide. Therefore, this model must be treated strictly as a decision-support tool requiring mandatory human oversight. To maintain public trust and avoid costly operational failures, future work must focus on data equity and model explainability to ensure transparent and responsible deployment in agricultural technology.

9 Conclusion and Future Work

This project successfully demonstrated that deep learning can automate the challenging task of vegetable leaf classification even when constrained by a small dataset. By comparing a custom model against transfer learning architectures, we achieved a definitive result. The **ResNet50 model**, utilizing residual connections and pre-trained ImageNet weights, proved to be the optimal solution, achieving the highest classification accuracy of **88.2%** on the independent test set. This confirms that leveraging powerful feature extraction capabilities learned from massive datasets is the most effective strategy for domain-specific problems characterized by data scarcity. Conversely, the poor performance of the InceptionV3 model underscored the importance of balancing network complexity with the amount of available training data. The study confirms that deep residual networks are particularly well-suited for agricultural vision tasks where the fine-grained discrimination of leaf texture and morphology is critical.

Building on this success, future work will focus on addressing the current limitations to move the framework closer to real-world deployment.

1. **Data Equity and Expansion:** The highest priority is correcting the class imbalance by collecting additional images for under-represented classes like Cabbage and Lettuce. This will eliminate bias and improve overall model fairness and reliability.

2. **Robustness via Background Removal:** We plan to implement **semantic segmentation** techniques to accurately isolate the leaf from the complex background (soil, shadows, mulch). This preprocessing step will force the classifier to focus solely on the plant features, which should significantly boost generalization accuracy in uncontrolled environments.

3. **Architectural Exploration:** We will investigate next-generation models, specifically **Vision Transformers (ViT)**. These attention-based architectures handle image patches differently than traditional CNNs and may provide superior performance by focusing on global leaf structure rather than just local pixel patterns.

References

[1] K. Too, L. Yujian, S. Njuki, and L. Ying, "A comparative study of fine-tuning deep learning models for plant disease identification," *Computers and Electronics in Agriculture*, vol. 161, pp. 272–279, 2019. [Online]. Available: `https://doi.org/10.1016/j.compag.2018.03.032`

[2] S. Sladojevic et al., "Deep neural networks based recognition of plant diseases by leaf image classification," *Computers and Electronics in Agriculture*, vol. 137, pp. 104–114, 2017. [Online]. Available: `https://doi.org/10.1016/j.compag.2017.02.002`

[3] A. Mokhtar, A. A. M. Azmi, and A. M. Shafie, "Plant leaf identification using deep learning and transfer learning," in *IOP Conference Series: Materials Science and Engineering*, vol. 917, 2020. [Online]. Available: `https://doi.org/10.1088/1757-899X/917/1/012015`

[4] P. Ferentinos, "Deep learning models for plant disease detection and diagnosis," *Computers and Electronics in Agriculture*, vol. 145, pp. 311–318, 2018. [Online]. Available: `https://doi.org/10.1016/j.compag.2018.01.009`

[5] M. Kaur, P. Kaur, and S. Gupta, "Identification of plant leaf diseases using CNN and transfer learning techniques," *Expert Systems with Applications*, vol. 208, p. 118111, 2022. [Online]. Available: `https://doi.org/10.1016/j.eswa.2022.118111`

[6] S. Rangarajan and R. Purushothaman, "Disease classification in eggplant leaves using transfer learning approaches," *Information Processing in Agriculture*, vol. 7, no. 4, pp. 585–593, 2020. [Online]. Available: `https://doi.org/10.1016/j.inpa.2020.01.003`