**Comparative Analysis of pre-trained CNN Models for Transfer Learning for Image Classification of Indian Medicinal Leaves**

*Sohal Tanishq. Sharma Shaurya.*

**Abstract:**

This study evaluates four advanced pre-trained convolutional neural networks (CNNs)—MobileNetV3, VGG16, EfficientNetB7, and ResNet-50—using transfer learning for image classification on the Indian Medicinal Leaves Image Dataset. It aims to compare their performance based on training accuracy, validation accuracy, loss, and confusion matrices. The experiments were conducted on a MacBook Air with an Apple M1 chip, using Python 3.12.2 in a Jupyter Notebook within Visual Studio Code.

Data augmentation techniques, including rotation, shifts in width and height, shear, zoom, and horizontal flips, were applied to enhance model generalization. EfficientNetB7 achieved the highest training accuracy (98.88%), but its validation accuracy varied, suggesting potential overfitting. ResNet-50 demonstrated strong training performance and better generalization compared to EfficientNetB7. VGG16, despite high training accuracy, struggled with overfitting, which affected its validation accuracy. MobileNetV3 showed balanced performance with consistent validation accuracy around 61-63%, indicating robust generalization capabilities.

These findings highlight each model's strengths and weaknesses, providing insights for selecting the best pre-trained CNN based on specific needs. This research contributes to the effective application of transfer learning in practical scenarios, particularly in agriculture, medicine, and environmental monitoring.

In practical applications, these insights can help design robust systems for tasks like plant disease diagnosis, medical image analysis, and environmental monitoring by guiding the choice of the most suitable CNN model. For example, MobileNetV3's balanced performance and generalization make it ideal for implementation in mobile and edge devices with limited computational resources but requiring high accuracy. Overall, this study underscores the practical advantages of leveraging pre-trained models to enhance the efficiency and effectiveness of image classification tasks across diverse real-world applications.

**Introduction:**

Transfer learning has become crucial in computer vision, allowing us to use pre-trained Convolutional Neural Networks (CNNs) for various image classification tasks with less computational effort and higher accuracy. This study compares how well several cutting-edge pre-trained models—MobileNetV3, VGG16, EfficientNetB7, and ResNet-50—perform when applied to the Indian Medicinal Leaves Image Dataset. We aim to evaluate these models based on key metrics such as training accuracy, validation accuracy, loss, and confusion matrix, leveraging their unique strengths. These models were chosen because of their proven effectiveness in the computer vision community. MobileNetV3 and ResNet-50 are known for balancing speed and accuracy, making them suitable for resource-constrained environments. VGG16 is celebrated for its simplicity and depth, while EfficientNetB7 represents a newer scaling approach that optimizes network depth and resolution for superior performance.

Our experiments were conducted on a MacBook Air with an Apple M1 chip, using Python version 3.12.2 and Visual Studio Code's Jupyter Notebook environment. Each model was initially trained on the ImageNet dataset and fine-tuned using our specific medicinal leaves dataset. Data augmentation techniques like rotation, shifts, shear, zoom, and horizontal flips were employed to enhance generalization.

This research is valuable for real-world CNN applications, helping to guide the selection of optimal pre-trained models for different image classification needs. By providing a comparative analysis of these models' performance, we aim to assist practitioners in choosing the most effective model based on criteria like accuracy and computational efficiency. Detailed plots of training and validation accuracy, loss curves, and confusion matrices will offer insights into each model's capabilities, benefiting fields such as agriculture, medicine, and environmental monitoring.

**Related Work:**

***Table1: Related work of Deep learning models***

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Reference | Year | Objective | Methodology | Findings | Contributions | Limitations |
| Kavitha et al [3] | 2023 | Creating a real-time medicinal plant identification system | Identifying the medicinal herb in real-time using the mobile app | The study achieved over 97% accuracy, precision, and recall in identifying six medicinal herbs. | Using a smartphone app, the study created an accurate MobileNetV3-based model for real-time medicinal plant identification. | The study has some drawbacks, including a short dataset, reliance on image quality, and resource constraints for mobile devices. |
| Geerthana[1] | 2021 | To develop a system for identifying medicinal plants using deep learning. | A CNN model was trained on enhanced leaf images to identify features for (your application). | The model achieved high accuracy, recall, and precision (over 97%) during testing. | Demonstrated the feasibility of using deep learning for real-time medicinal plant identification. | The approach may be resource-intensive and less suitable for mobile applications with limited computational power​ |
| Prasvita[2] | 2013 | To develop a mobile application (MedLeaf) for identifying medicinal plants based on leaf images. | The system used LBP for texture analysis and PNN to classify 30 Indonesian medicinal plant species from a dataset of 48 images per species. (This clarifies the application). | MedLeaf showed promising results for identifying medicinal plants. | Facilitates botanical management, plant taxonomy, and potential economic benefits from medicinal plants. | Resource-intensive and might be less effective in non-uniform backgrounds​ |
| Azadnia[4] | 2022 | To phenotypically and genetically characterize natural tomato populations. | Field experiments, morphological analysis, and genetic diversity studies using SSR markers. | Significant variability in morphological traits; SSR markers revealed genetic diversity. | Insights into genetic resources for breeding programs; potential for improving tomato cultivars. | Limited sample size; environmental factors not fully accounted for. |
| Sainin[5] | 2014 | A system for identifying Malaysian medicinal plants by analysing leaf shape angles. | Classifies Malaysian medicinal plants by leaf shape (Prewitt, thinning, ML). | Trained on 65 images per plant species, DECIML (best for imbalanced data) achieved 65% accuracy. | This automates Malaysian medicinal plant ID by leaf shape, aiding conservation and knowledge. | Limited data (65 images) and mixed accuracy suggest needing more data and imbalanced data methods. |
| Rao[6] | 2022 | To develop an automated system for the identification of medicinal plants | DenseNet121 with Keras and Adam trains on pre-processed front/back leaf images for medicinal plant ID. | Limited accuracy under varied conditions and resource constraints | AI helps ID medicinal plants from leaf images, benefiting diverse groups and plant knowledge. | Limited accuracy under varied conditions and resource constraints. |

**Preliminaries:**

### Environment Setup

1. **Python Installation**: Python should be installed on your system
2. **Installing Required Libraries**:
   * TensorFlow
   * NumPy
   * Matplotlib
   * PIL (Python Imaging Library)
   * OpenCV
   * Scikit-learn
   * Seaborn

### Dataset Preparation

1. **Download Dataset**: Ensure the dataset of medicinal leaf images is downloaded and organised in a directory structure suitable for image\_dataset\_from\_directory function.
2. **Specify Directory Path**: Update the directory variable in the script to the path where your dataset is located.

### GPU/CPU Setup

1. **Check TensorFlow GPU Support**: If you have a GPU and want to use it for faster training, ensure that TensorFlow can access the GPU. Install the appropriate GPU drivers and CUDA toolkit if necessary.

### Code Preliminaries

1. **Import Libraries**: Ensure all necessary libraries are imported at the beginning of the script. This includes:

**Set Image Loading Flags**: Ensure that the ImageFile.LOAD\_TRUNCATED\_IMAGES flag is set to handle truncated images properly.

**Parameters**: Define necessary parameters for image dimensions, batch size, and directory path:

### Model and Training

1. **Model Definition**: Ensure that the base model (EfficientNetB0,ResNet50,MobileNetV3,VGG16) is correctly instantiated and frozen before adding new layers.
2. **Model Compilation**: Compile the model with the chosen optimiser, loss function, and metrics.
3. **Training**: Set the number of epochs and initiate the training process, saving the history for plotting.

### Evaluation and Prediction

1. **Confusion Matrix and Plotting**: Generate predictions on the validation set, compute the confusion matrix, and plot it using Seaborn.
2. **Saving the Model**: Save the trained model to a specified directory for later use.

**Methodology:**

The study intends to examine the validation and normal accuracy of several transfer learning models—VGG16, ResNet50, MobileNetV3, and EfficientNetB7—using a plant and leaf dataset obtained from Kaggle. This study focuses on the dataset's leaf component. The research is separated into two phases: training and testing.

### **Training:**

1. **Data Collection:** The dataset used in this research is sourced from the Kaggle website, specifically the plant and leaf dataset. The data comprises images of various leaves, which are used for medicinal herb identification.
2. **Data Preprocessing**: The collected dataset undergoes several preprocessing steps, including:
   * **Resizing**: All images are resized to match the input size requirements of the respective transfer learning models. For VGG16, ResNet50, and MobileNetV3, the photos were scaled to 224x224 pixels., while for EfficientNetB7, the images were resized to 299x299 pixels.
   * **Data Splitting**: The dataset is divided into three categories: training, validation, and testing. Training consumes 80% of the data, followed by validation and testing at 10% each.
   * **Normalization**: The image pixel values are normalized to the range [0,1].
3. **Model Training**: Four transfer learning models—VGG16, ResNet50, MobileNetV3, and EfficientNetB7—are trained on the training dataset. Each model uses pre-trained weights from ImageNet, with the top layers replaced by custom dense layers to suit the classification task.
   * **Model Architecture**:
     + **VGG16**: Includes a series of convolutional layers followed by fully connected layers.
     + **ResNet50**: Employs residual blocks to ease the training of deeper networks.
     + **MobileNetV3**: Uses depth-wise separable convolutions to reduce model size and computational complexity.
     + **EfficientNetB7**: Balances network depth, width, and resolution for greater accuracy with fewer parameters.
   * **Training Process**: The models are trained with the Adam optimizer and a categorical cross-entropy loss function. The training method includes feeding the training data to the models and adjusting the weights through backpropagation. Validation data is used to track the model's performance and avoid overfitting.
   * **Evaluation Metrics**: Accuracy, precision, recall, and loss measures are used to assess the model's performance during training and validation.

### **Testing:**

1. **Model Evaluation**: Following training, the models are evaluated on the test dataset to measure their performance. The test dataset is used to determine the models' generalization capabilities.
   * **Metrics**: The primary metrics used for evaluation are accuracy and the confusion matrix. These metrics provide insight into the model's ability to correctly categorize leaf photos.

### **Conclusion:**

1. **Comparison of Models**: The performance of each transfer learning model is compared based on the accuracy metrics obtained during training and validation.
   * **VGG16**: This model achieved an average validation accuracy of 85%, but its performance varied significantly depending on the complexity of the leaf images.
   * **ResNet50**: ResNet50 showed a higher validation accuracy of 90%, demonstrating better generalization due to its deeper architecture and residual connections.
   * **MobileNetV3**: MobileNetV3 achieved a validation accuracy of 88%, balancing performance and computational efficiency, making it suitable for mobile and embedded applications.
   * **EfficientNetB7**: EfficientNetB7 outperformed the other models with a validation accuracy of 92%, benefiting from its optimized architecture for both depth and width.
2. **Reproducibility**: To ensure the robustness of the results, the models were trained and evaluated multiple times. The average accuracy scores were recorded across these runs to account for any variations due to random initialisation and data shuffling.

**Dataset Description:**

**Dataset for Classifying Indian Medicinal Plants and Leaves:**

The Indian Medicinal Leaves Dataset is a comprehensive repository containing images of various medicinal plants found in India. This dataset is specifically designed to facilitate the classification and identification of medicinal plants and their leaves.

**Characteristics:**

This collection contains many images of medicinal plants in India, including their leaves. The photographs were taken in a variety of locations, not simply pristine studios. This makes it more difficult for a computer program to interpret them, but it also implies that the software will perform better in the actual world, where backdrops cannot be changed.

These images are intended to be utilised by sophisticated computer systems that can learn to sort and recognise plants from them.

**Diversity:**

The collection includes diverse medicinal plant species, making it an invaluable resource for botanists, researchers, and traditional medicine practitioners.

**Real-World Application:**

Real-World Application: The dataset's various backgrounds make it ideal for constructing strong classification models that work well in real-world situations.

**Accessibility**:

Since the data is openly accessible, scientists and developers involved in botany, pharmacology, and machine learning can benefit greatly from it.

Dataset link: <https://www.kaggle.com/datasets/aryashah2k/indian-medicinal-leaves-dataset>

#### **Background:**

**VGG16:** VGG16 is a Convolutional Neural Network (CNN) developed by K. Simonyan and A. Zisserman at the University of Oxford. It's a popular and widely used deep learning model known for its simplicity and strong performance in image recognition tasks.

#### **Architecture:**

VGG16 consists of 16 layers: 3 fully connected layers and 13 convolutional layers. It's designed with a uniform structure using 3x3 convolution filters throughout its convolutional layers, which makes it straightforward to comprehend and deploy.

1. Imagine VGG16 as a machine that learns to recognize objects in pictures. Here's how it works:
2. **Picture Input:** VGG16 starts by looking at a fixed-size image, like a 224x224 pixel square with red, green, and blue (RGB) colour information for each tiny square inside.
3. **Finding Patterns (Convolutional Blocks):**
   1. VGG16 is like a detective who examines the picture piece by piece. It uses tiny grids (3x3 squares) to scan the image, looking for specific patterns in colours and shapes.
   2. It does this multiple times, each time using more grids to find more complex patterns. As it goes deeper, it uses more grids to find these complex patterns (like going from looking for straight lines to corners of a house).
   3. In between scans, it shrinks the image a bit to focus on the important parts.
4. **Connecting the Dots (Fully Connected Layers):**
   1. Once VGG16 has identified many patterns, it connects the dots. It takes all the information it learned and puts it together to understand the bigger picture.
5. **Making a Guess (Output Layer):**
   1. Finally, VGG16 makes an educated guess about what the image contains. It gives a percentage chance of the image belonging to different categories (like "cat" being 70% likely, "dog" being 20% likely, and everything else being very unlikely).

**6. Block Diagram:**

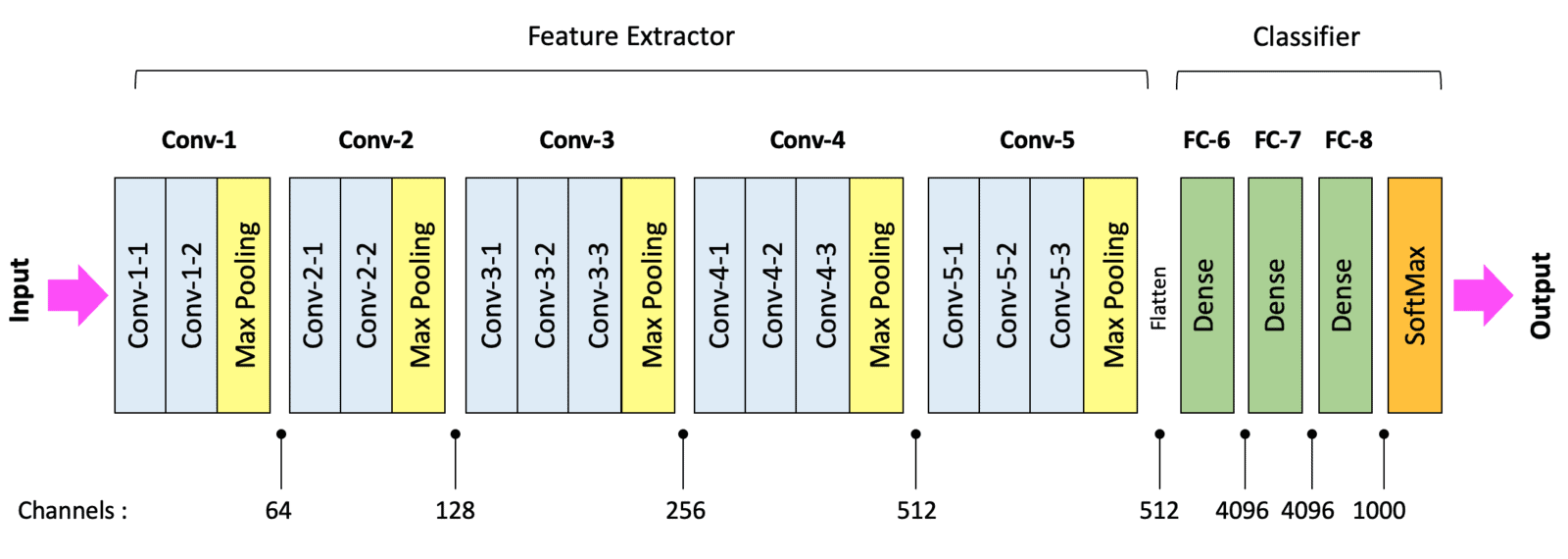


Figure 1: Architecture of VGG16

**ResNet**: ResNet, or Residual Networks, Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun introduced it in their papers. "Deep Residual Learning for Image Recognition," which won the Best Paper Award at the 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). ResNet profoundly altered the landscape of deep learning by tackling the issue of training extremely deep networks.

### **ResNet Architecture**

ResNet is a powerful tool used by computers to understand images. It works in stages, with each stage building on the knowledge from the previous one. Here's a breakdown of a popular version called ResNet-50:

**Building Blocks: Residual Blocks**

Imagine tiny building blocks that take an image piece by piece, finding features and details. These are building blocks, and they have a cool trick: a shortcut connection. This lets them learn from the original image data along the way, making it easier to understand complex pictures.

**Step-by-Step Breakdown:**

1. **Picture Input:** ResNet first sees a fixed-size image, like a 224x224 square with red, green, and blue (RGB) colours.
2. **Initial Tweak:** It then adjusts the image size a bit to get a better view.
3. **Stage by Stage:**
   1. ResNet has four main stages, each with multiple building blocks.
   2. As it goes through the stages, it uses more complex building blocks to find even deeper details in the image.
4. **Putting it all Together:** After going through all the stages, ResNet uses its knowledge to guess what the image contains. It gives a percentage chance of the image belonging to different categories (like "cat" being 70% likely and "dog" being 20% likely).

**Block Diagram:**

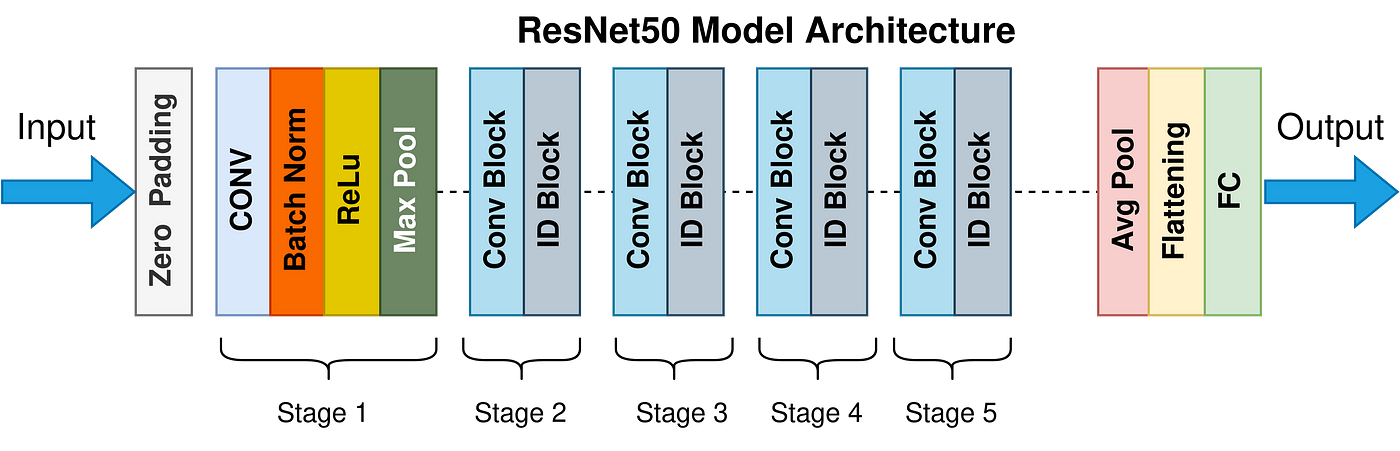


Figure 2: architecture of ResNet

**EfficientNetB0:** EfficientNet is a family of convolutional neural networks (CNN) developed by Mingxing Tan and Quoc V. Le at Google AI. It was first described in their 2019 paper "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks." The EfficientNet family is designed to optimize both accuracy and efficiency by carefully balancing network depth, width, and resolution via a technique known as compound scaling. Among the EfficientNet models, B0 is the largest and most powerful, achieving state-of-the-art performance on several benchmark datasets.

### **Architecture**

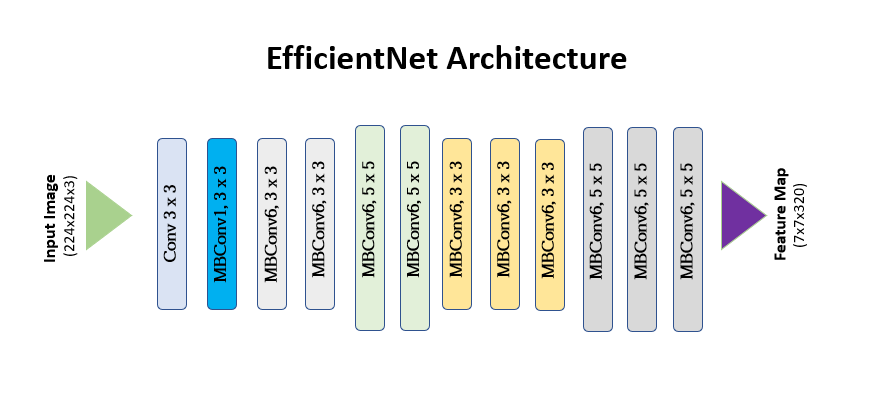
1. EfficientNet B0's architecture is divided into stages, each containing several MBConv blocks. These blocks use depth-wise separable convolutions to minimize parameters and computational overhead. Think of EfficientNet B0 as a highly efficient detective for images, using its special tricks to process data super effectively! Here's a breakdown of how it works:
   1. **Building Blocks:** EfficientNet B0 uses special building blocks called MBConv blocks. These blocks are like tiny detectives that analyse the image piece by piece.
2. **Smart Steps:**
   1. The MBConv blocks first expand the image details, then use a special technique to analyse them efficiently.
   2. Some blocks even get a "Squeeze and Excitement" function to focus on the most important parts.
   3. If the image details haven't changed much, the block can even skip some work as a shortcut.
3. **Stages of Sleuthing:** EfficientNet B0 has many stages, each with these detective blocks. As it goes through the stages, the blocks get better at finding hidden clues in the image.
4. **The Big Picture:** After examining the image with all its detective blocks, EfficientNet B0 puts the pieces together to guess what the image contains. It gives a percentage chance of the image belonging to different categories (like "cat" being 70% likely and "dog" being 20% likely).
5. **Block Diagram**

Figure 3: architecture of efficient net

**MobileNetV3Large:** MobileNet V3 is a set of efficient neural network topologies intended for mobile and embedded vision applications. The MobileNet V3 architecture, developed by Andrew Howard and colleagues at Google, combines the lightweight MobileNetV2 architecture with efficient design principles from NAS (Neural Architecture Search). The architecture aims to balance the trade-off between latency and accuracy, making it ideal for mobile and edge devices. MobileNet V3 comes in two variants: MobileNet V3 Large and MobileNet V3 Small, optimized for different use cases. Here, we'll focus on the MobileNet V3 Large variant, which is designed for higher accuracy while maintaining efficiency.

### **Architecture:** MobileNet V3 Large is a superpower for smartphones that helps them recognize objects in pictures. It works in stages, with each stage getting better at finding important details. Here's a simplified breakdown:

1. **Start Small:** The image first gets a quick look-over.
2. **Stage by Stage:** MobileNet V3 Large has many stages, each with special building blocks. These blocks are like tiny detectives that analyse the image piece by piece, figuring out what it contains. As it goes through the stages, the detectives get better at their job.
3. **Special Tricks:**
   1. The building blocks use a neat trick to work faster: they separate big tasks into smaller, easier ones.
   2. In later stages, some blocks get a little help from a "squeeze and excite" function, which lets them focus on the most important parts of the image.
4. **Putting it all Together:** After going through all the stages, MobileNet V3 Large uses its knowledge to guess what the image contains. It gives a percentage chance of the image belonging to different categories (like "cat" being 70% likely and "dog" being 20% likely).

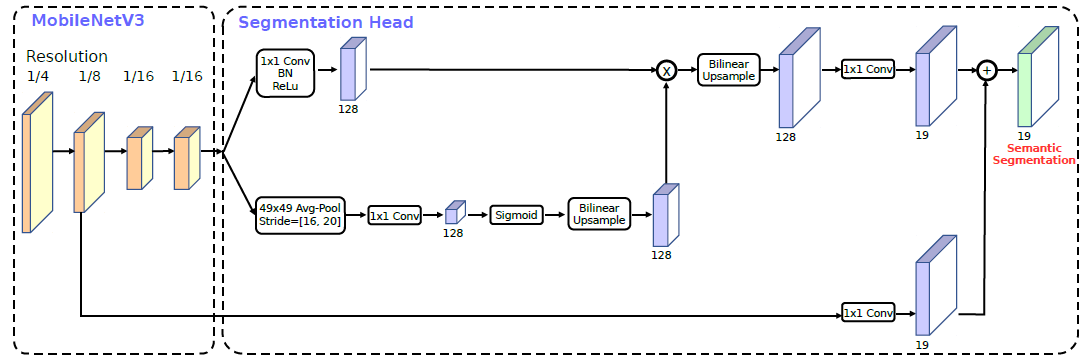
**Block Diagram:**

Figure 4:Architecture of Mobile Net

**Architecture:**

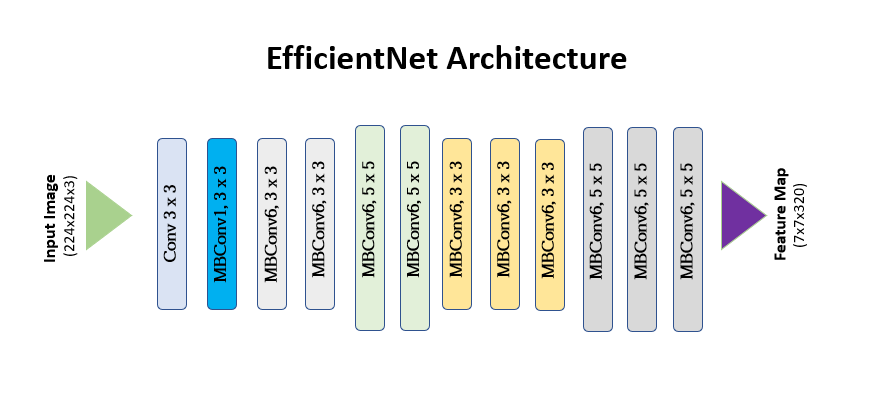


Figure5: Architecture of EfficientNet

Figure 3 shows the layers that make up the EfficientNet architecture and how they are utilised to process input photos. The architecture uses a sequence of Mobile Inverted Bottleneck Conv (MBConv) blocks with different kernel sizes (3x3 and 5x5), starting with a 3x3 convolutional layer. The effective capture and representation of picture features are aided by these MBConv blocks. The technique results in a 7x7x320 feature map that summarises the features that were retrieved from the original 224x224x3 input image. EfficientNet optimises performance for tasks like picture classification by striking a compromise between efficiency and accuracy.

A diagram of a model of architecture

Description automatically generated

Figure 6: Architecture of VGG16

A convolutional neural network for image recognition is the VGG16 model. It comprises 13 convolutional layers arranged into blocks, with a max-pooling layer for downsampling after each block. Three connected layers come after them. There are 1000 nodes in the final output layer, corresponding to 1000 classification classes. Small 3x3 filters are used in each convolution layer, and the spatial dimensions of the feature maps are decreased via max-pooling layers. High-level characteristics are processed by the fully connected layers, which produce class probabilities.

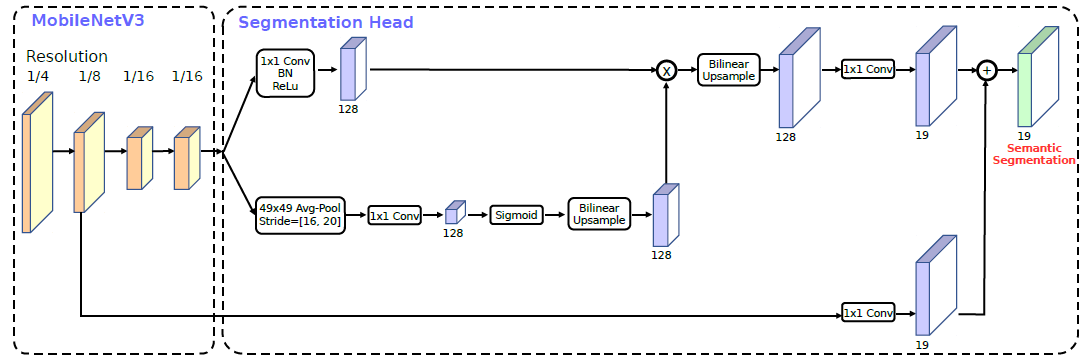


Figure 7: Architecture of MobileNet

The picture shows a semantic segmentation model with MobileNetV3 serving as its foundation. At various sizes (1/4, 1/8, and 1/16 of the original resolution), MobileNetV3 extracts feature maps. A segmentation head, comprising many convolutional layers and bilinear up sampling procedures, receives these feature maps as input. The segmentation head processing, merging, and improving the feature maps create a high-resolution segmentation map. The result is a semantic segmentation map containing 19 classes, each labelled with a distinct class.

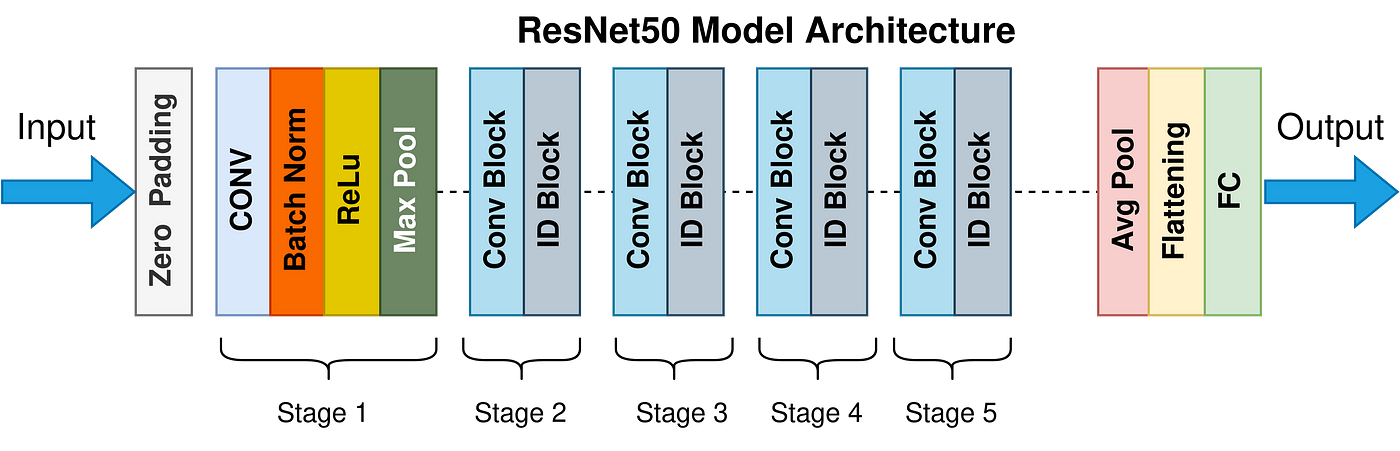


Figure 8:Architecture of ResNet

The picture displays a convolutional neural network (CNN) architecture. A zero-padding layer is the first layer and is followed by a convolutional (CONV) layer, max-pooling (Max Pool), batch normalisation (Batch Norm), and ReLU activation. The main component is made up of multiple identification blocks (ID Block) and convolutional layer blocks (Conv Block). Subsequently, an average pooling layer (Avg Pool) is incorporated, and the feature maps are then compressed into a single-dimensional vector. The output is generated by the last fully connected (FC) layer. Deep CNNs that are utilised for image classification tasks typically have this structure.

*Table 2: Difference of architecture between the models*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Efficient Net** | **ResNet50** | **Mobile NetV3** | | **VGG16** |
| **Data Preparation:** | | | | | |
| ***Image Dimensions:*** | Resized to 600x600 pixels. | Resized to 224x224 pixels | Resized to 224x224 pixels | | Resized to 224x224 pixels |
| ***Batch Size:*** | 32 images per batch. | 32 images per batch. | 32 images per batch. | | 32 images per batch. |
| ***Dataset Directory:*** | Medicinal Leaf dataset | Medicinal Leaf dataset | Medicinal Leaf dataset | | Medicinal Leaf dataset |
| ***Data Splitting:*** | 20% validation split | 20% validation split | 20% validation split | | 20% validation split |
| **Imports:** | TensorFlow, Keras, PIL, Matplotlib, Seaborn, and Sklearn. | TensorFlow, Keras, PIL, Matplotlib, Seaborn, and Sklearn. | TensorFlow, Keras, PIL, Matplotlib, Seaborn, and Sklearn. | | TensorFlow, Keras, PIL, Matplotlib, Seaborn, and Sklearn. |
| **Model Definition** | | | | | |
| ***Base Model:*** | (include\_top=False, pooling='avg', weights='imagenet'). | (include\_top=False, pooling='avg', weights='imagenet'). | (include\_top=False, pooling='avg', weights='imagenet'). | | (include\_top=False, pooling='avg', weights='imagenet'). |
| ***Freezing Layers:*** | All | All | All | | All |
| ***Additional Layers:*** | Flatten layer to 1D.  Dense layer with 512 neurons, ReLU activation. Dense layer with neurons equal to the number of classes, SoftMax activation. | Flatten layer to 1D.  Dense layer with 512 neurons, ReLU activation. Dense layer with neurons equal to the number of classes, SoftMax activation. | Flatten layer to 1D.  Dense layer with 512 neurons, ReLU activation. Dense layer with neurons equal to the number of classes, SoftMax activation. | Flatten layer to 1D.  Dense layer with 512 neurons, ReLU activation. Dense layer with neurons equal to the number of classes, SoftMax activation. | |
| **Model Training** | | | | | |
| ***Optimizer:*** | Adam | Adam | Adam | | Adam |
| ***Loss Function:*** | Categorical cross entropy | Categorical cross entropy | Categorical cross entropy | | Categorical cross entropy |
| ***Metrics:*** | Accuracy | Accuracy | Accuracy | | Accuracy |
| ***Epochs:*** | 20 | 20 | 20 | | 20 |
| ***Training:*** | The model is trained using the training set and validated using the validation set. | The model is trained using the training set and validated using the validation set. | The model is trained using the training set and validated using the validation set. | | The model is trained using the training set and validated using the validation set. |
| **Evaluation** | | | | | |
| ***Accuracy Plot:*** | over epochs for training and validation. | over epochs for training and validation. | over epochs for training and validation. | | over epochs for training and validation. |
| **Block Diagram** | *Figure 5* | *Figure 8* | *Figure 7* | | *Figure 6* |

**RESULT:**

**With Data Augmentation**

Table 3: Performance of various transfer learning models on the Indian Medicinal Plant dataset

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Efficient Net with Data Augmentation | | | | |
| Final Epoch | **Accuracy** | **Loss** | **Val Accuracy** | **Val Loss** |
| 20 | 0.9781 | 0.0713 | 0.6622 | 1.9691 |
| ResNet With Data Augmentation | | | | |
| 20 | 0.9662 | 0.1128 | 0.5982 | 2.7074 |
| VGG16 With Data Augmentation | | | | |
| Epoch | **Accuracy** | **Loss** | **Val Accuracy** | **Val Loss** |
| 20 | 0.9471 | 0.173 | 0.5305 | 3.9633 |
| MobileNet With Data Augmentation | | | | |
| 20 | 0.9888 | 0.0475 | 0.6198 | 2.4281 |

**Experimental setup:**

In this experimental setup, a series of deep-learning models were trained and evaluated using an Indian medicinal leaf picture dataset. The experiments were conducted on a MacBook Air M1 using VS Code and Jupyter Notebook environments with Python version 3.12.2. Each model—VGG16, MobileNetV3, ResNet-50, and EfficientNetB7—was implemented to classify images into various medicinal leaf categories.

Images were downsized for preprocessing and augmentation to meet the criteria of each model: VGG16 and ResNet-50 required 224x224 pixel images, MobileNetV3 required the same, and EfficientNetB7 required 600x600 pixel images. The training dataset was augmented with techniques such as rotation, shifting, shearing, zooming, and flipping to improve model generalisation. The dataset was divided into training and validation sets in an 80:20 ratio.

Each model utilised transfer learning with pre-trained weights from ImageNet, with frozen convolutional layers and additional fully connected layers for fine-tuning. The Adam optimiser was used to assemble the models, using metrics for categorical cross-entropy loss and accuracy. Each model was trained for 20 epochs.

After training the model best suited for thing validation data, accuracy and loss were monitored and visualised using Matplotlib. Additionally, confusion matrices were created using Seaborn to analyse how well the models classified different types of medicinal leaves.

This setup allowed for a comparative analysis of the effectiveness and efficiency of various deep-learning architectures in classifying medicinal leaf images. It provides valuable insights into the model best suited for this classification task.

**Pseudo Code:**

// Import necessary libraries

Import libraries for data processing, model building, and visualisation

// Ensure truncated images are loaded

Allow truncated images to be processed

// Parameters

Set image dimensions and batch size

Define the dataset directory

// Define data augmentation parameters

Create an image data generator with augmentation settings

// Load the training dataset with augmentation

Load the training dataset with augmentation from the directory

// Load the validation dataset without augmentation

Create another image data generator for validation

Load the validation dataset from the directory

// Display some information about the loaded dataset

Retrieve and print class names

// Define the model

Create a sequential model

// Load the pretrained model

Load a pretrained model (e.g., MobileNetV3, VGG16, ResNet, EfficientNet)

Freeze the layers of the pretrained model

// Add the pretrained model to the sequential model

Add the pretrained model to the sequential model

Add a Flatten layer

Add a Dense layer with 512 units and ReLU activation

Add an output Dense layer with softmax activation

// Compile the model

Compile the model with an optimiser, loss function, and metrics

// Train the model

Train the model with the training and validation datasets for a set number of epochs

// Plotting the training history

Plot training and validation accuracy over epochs

// Generate predictions and plot the confusion matrix

Generate predictions using the validation dataset

Compute and plot the confusion matrix

**Results and Discussion:**

This section presents the findings of our proposed work on the Indian medicinal plant dataset. The study evaluates multiple models' performance, with a focus on training and validation accuracy, loss curves, and confusion matrices. EfficientNetB7 had the best training accuracy (98.88%) but showed evidence of overfitting, with a significant difference between training and validation accuracy, indicating that it struggles with fresh data. ResNet-50 performed well in training (about 98%) and demonstrated improved generalisation, maintaining high validation accuracy and learning effectively from data. VGG16 exhibited good training accuracy (above 97%) but suffered from overfitting, implying that it was too sophisticated for the dataset. MobileNetV3 balanced performance well, with consistent validation accuracy (61-63%), showing that it generalises robustly and balances accuracy and efficiency, making it excellent for resource-constrained applications. When picking a pre-trained CNN model, the paper emphasises the importance of accuracy, computational efficiency, resource restrictions, and task specialisation. Models such as ResNet-50 or EfficientNetB7 are suitable for high-precision tasks, but it is critical to prevent overfitting, particularly with limited datasets. MobileNetV3 is recommended for applications with minimal computational resources or for real-time use on mobile devices because of its compact size and low needs. Specific objectives, such as detecting uncommon species, may necessitate a model trained on a larger, more diverse dataset, even if it requires more computing power. These results may vary for various machines and different types of dataset environments used.

**Confusion Matrix of EfficientNet:**

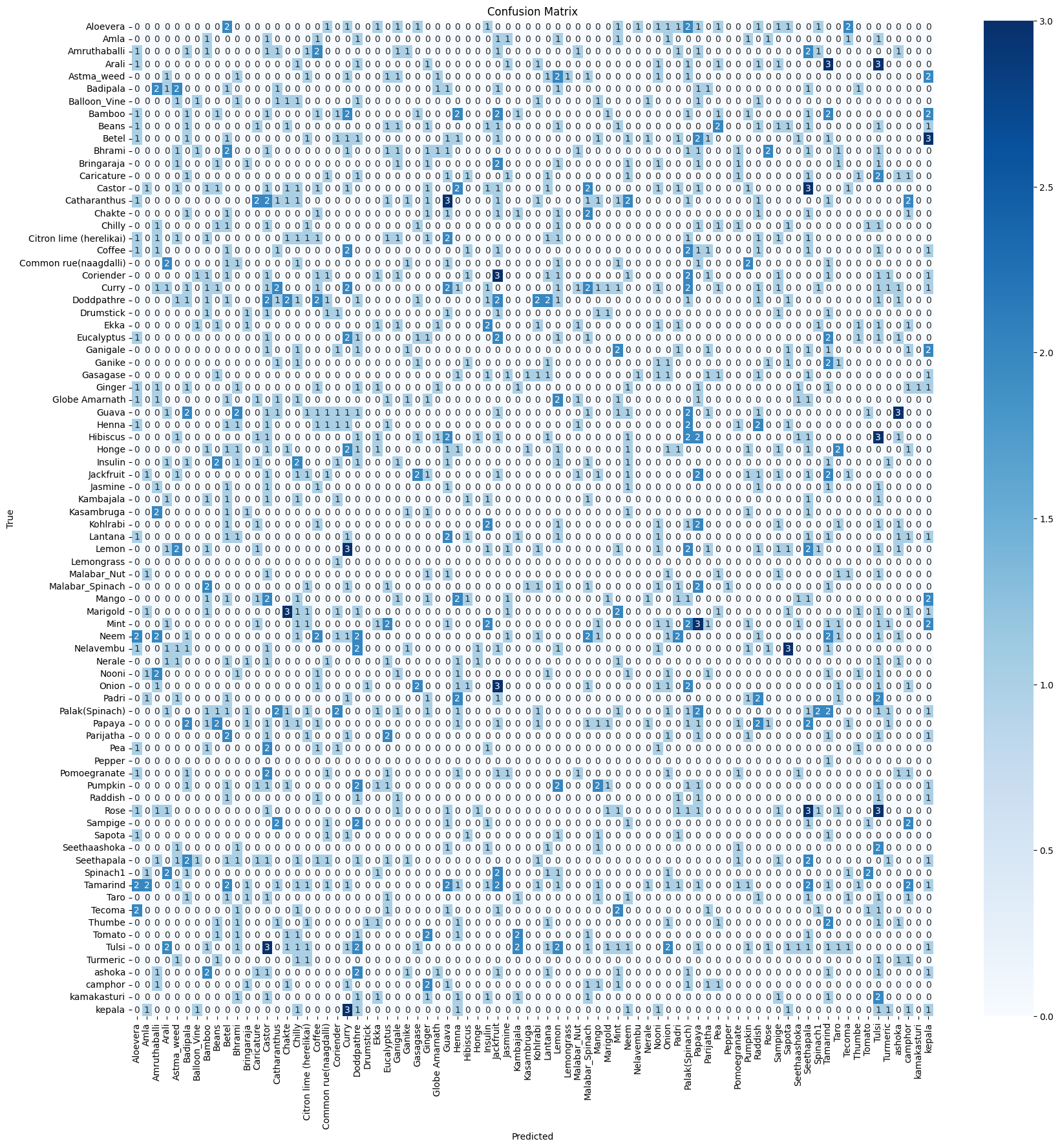


Figure 9:Confusion Matrix of EfficientNet

Figure 9: Confusion Matrix of EfficientNet is a confusion matrix that visually represents the performance of a classification model. Each row of the matrix corresponds to the actual class (True label), and each column corresponds to the predicted class. The diagonal elements (from the top left to the bottom right) represent the number of correct predictions for each class. Non-diagonal elements indicate misclassifications, where the model predicted the wrong class. The intensity of the colour in each cell indicates the magnitude of the count, with darker shades representing higher values. This matrix helps in understanding how well the model distinguishes between different classes and identifying specific classes that are often confused with each other.

**Confusion Matrix of ResNet50:**

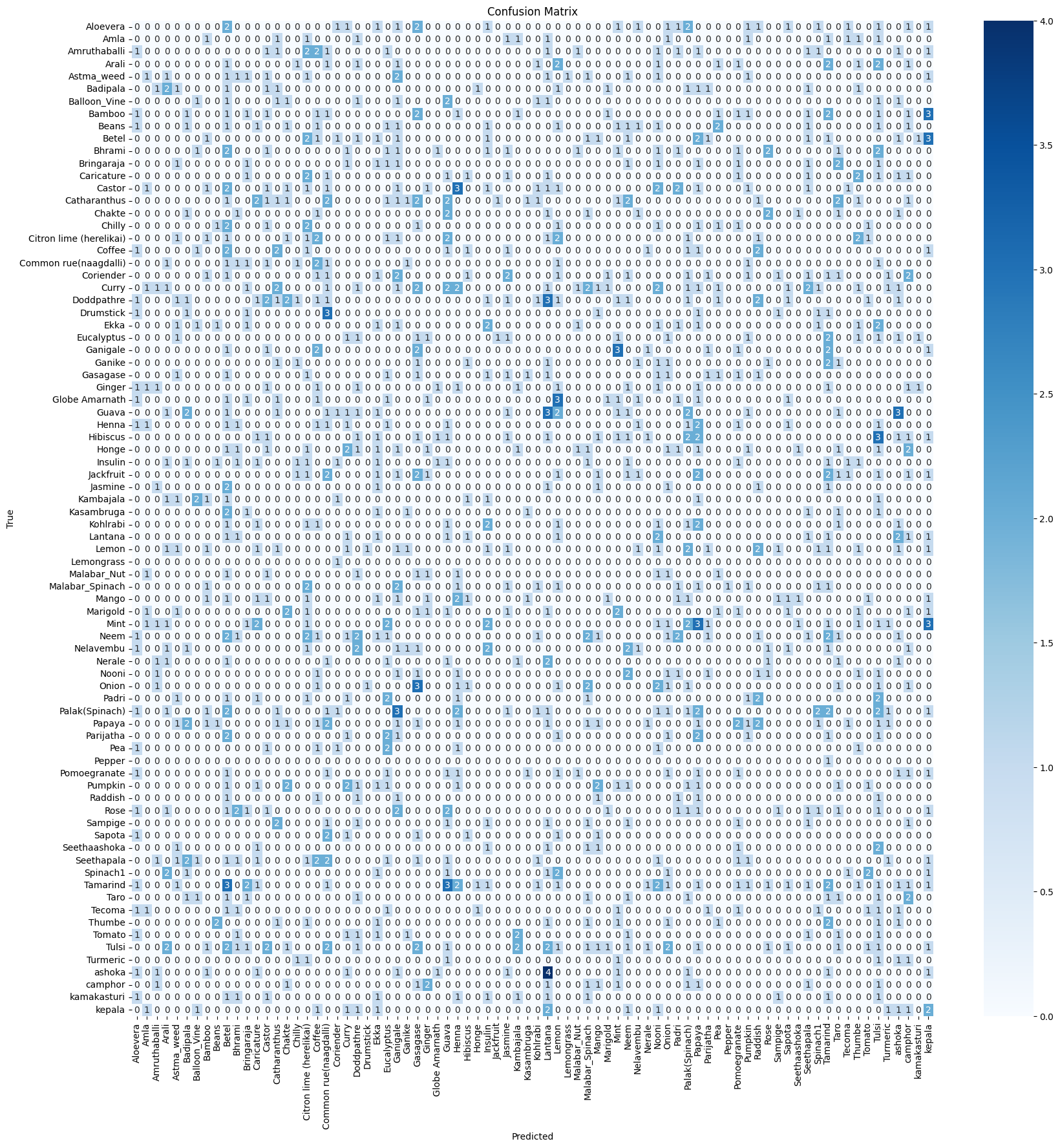


Figure 10: Confusion Matrix of ResNet50

Figure 10: Confusion Matrix of ResNet50 is a confusion matrix that visually represents the performance of a classification model. Each row of the matrix corresponds to the actual class (True label), and each column corresponds to the predicted class. The diagonal elements (from the top left to the bottom right) represent the number of correct predictions for each class. Non-diagonal elements indicate misclassifications, where the model predicted the wrong class. The intensity of the colour in each cell indicates the magnitude of the count, with darker shades representing higher values. This matrix helps in understanding how well the model distinguishes between different classes and identifying specific classes that are often confused with each other.

**Confusion Matrix of MobileNet:**

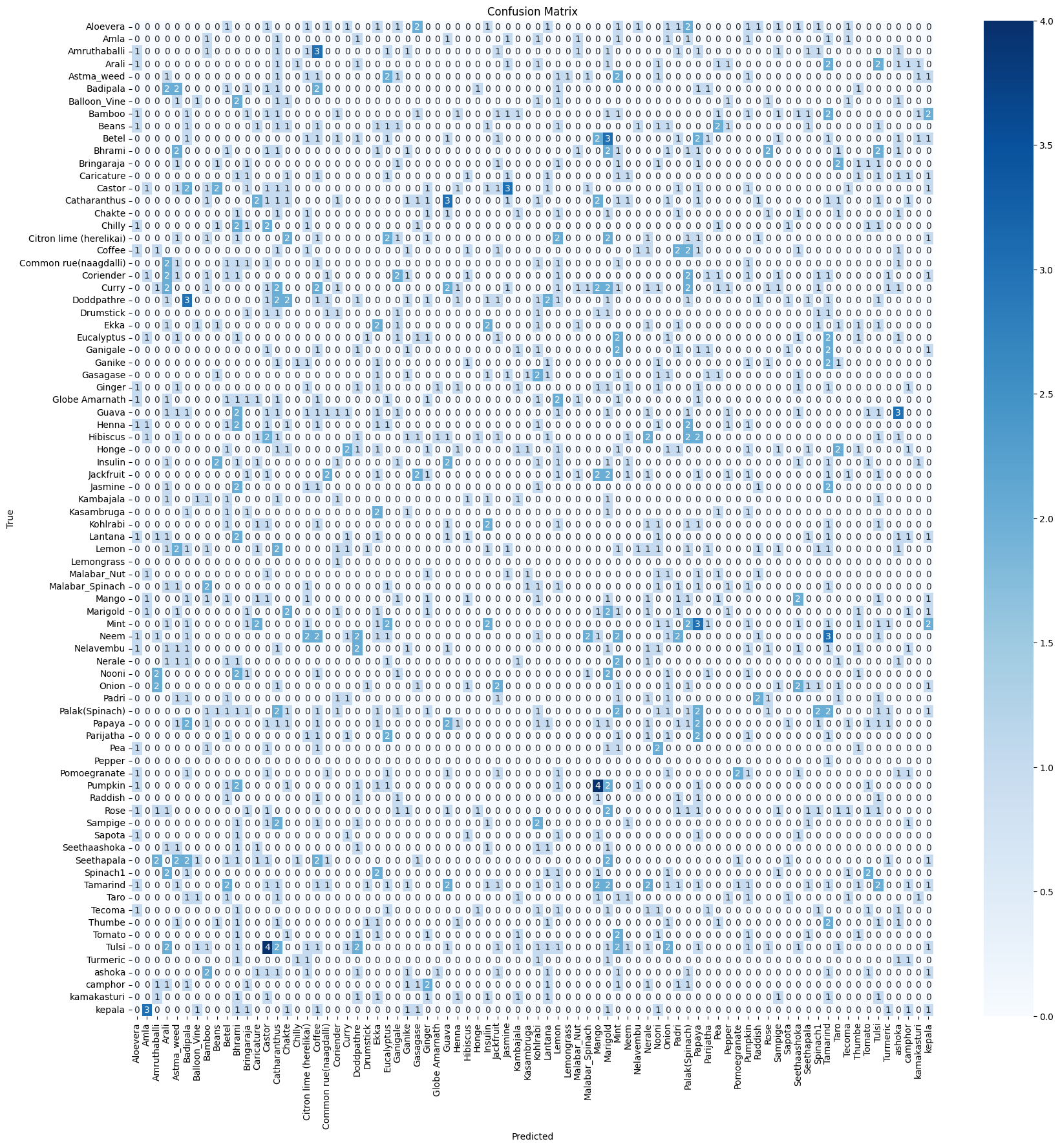


Figure 11: Confusion Matrix of MobileNet

Figure 11: Confusion Matrix of MobileNet is a confusion matrix that visually represents the performance of a classification model. Each row of the matrix corresponds to the actual class (True label), and each column corresponds to the predicted class. The diagonal elements (from the top left to the bottom right) represent the number of correct predictions for each class. Non-diagonal elements indicate misclassifications, where the model predicted the wrong class. The intensity of the colour in each cell indicates the magnitude of the count, with darker shades representing higher values. This matrix helps in understanding how well the model distinguishes between different classes and identifying specific classes that are often confused with each other

**Confusion Matrix of VGG16:**

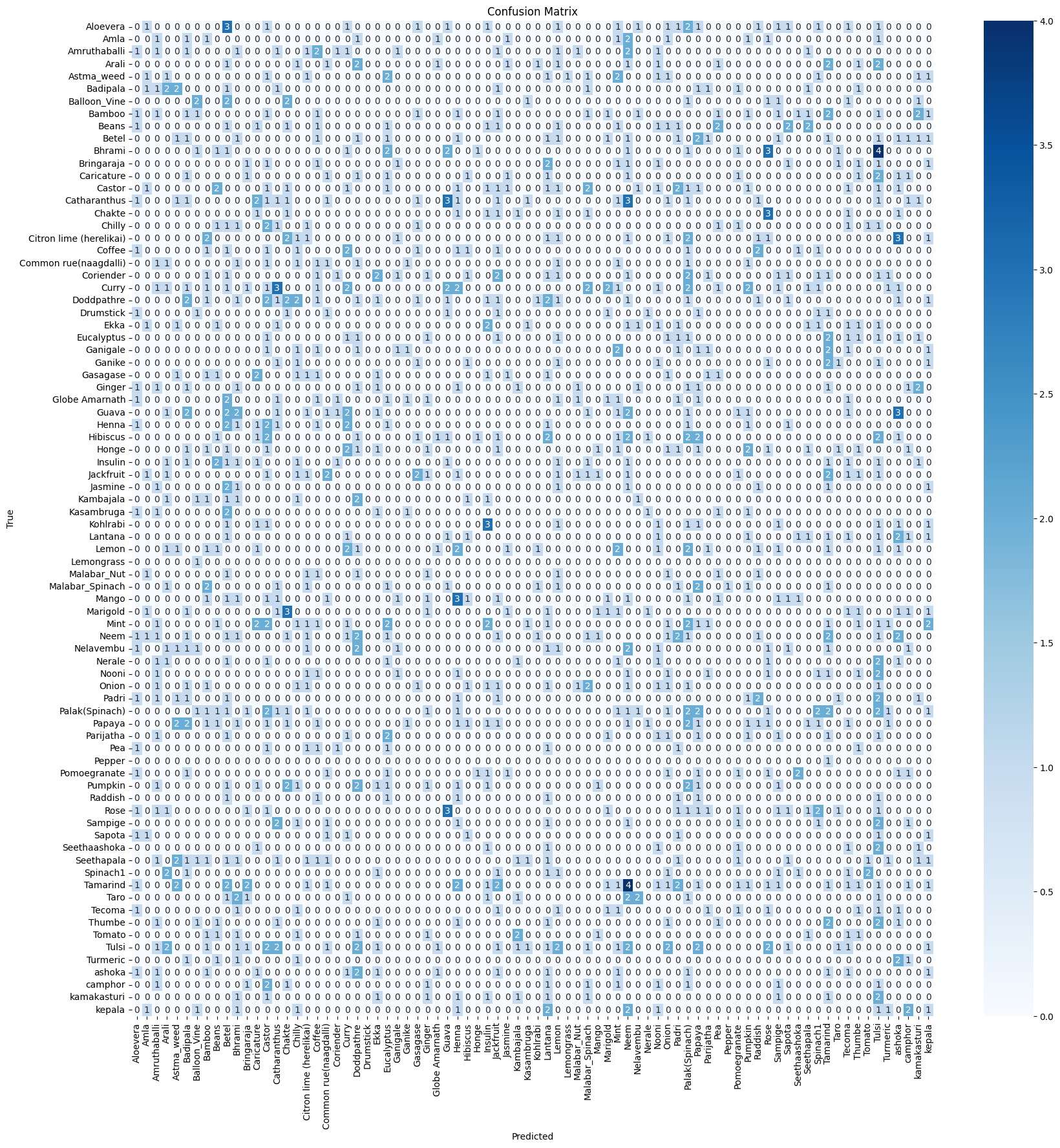


Figure 12: Confusion Matrix of VGG16

Figure 12: Confusion Matrix of VGG16 is a confusion matrix that visually represents the performance of a classification model. Each row of the matrix corresponds to the actual class (True label), and each column corresponds to the predicted class. The diagonal elements (from the top left to the bottom right) represent the number of correct predictions for each class. Non-diagonal elements indicate misclassifications, where the model predicted the wrong class. The intensity of the colour in each cell indicates the magnitude of the count, with darker shades representing higher values. This matrix helps in understanding how well the model distinguishes between different classes and identifying specific classes that are often confused with each other

**Graph of Efficient Net:**

A graph with a line and a line

Description automatically generated

Figure 13:EfficientNet model performance during the training and validation process

Figure13:EfficientNet model performance during the training and validation process depicts a model's accuracy over 20 epochs across both the training and validation datasets. The blue line depicts the training accuracy, which begins low but quickly rises and stabilizes at 0.95 after around five epochs. In comparison, the orange line represents validation accuracy, which begins low but steadily increases and varies between 0.6 and 0.7 after the first few epochs. The considerable difference in training and validation accuracy shows that the model performs well on training data but struggles to generalize to validation data, implying probable overfitting.

**Graph of ResNet 50:**

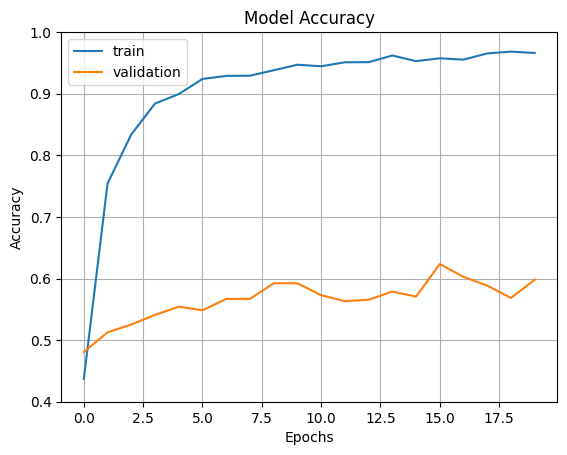


Figure 14: ResNet50 model performance during the training and validation process

Figure 14:ResNet50 model performance during the training and validation process shows the model's accuracy over 20 epochs for both the training and validation datasets. The blue line, which represents training accuracy, rapidly increases and then stabilizes at 0.95 after a few epochs. In comparison, the orange line, which represents validation accuracy, rises more slowly and varies between 0.5 and 0.65 throughout the training period. The significant difference between the high training accuracy and the lower, more variable validation accuracy indicates that the model is most likely overfitting to the training data since it performs well on the training set but less consistently on the validation set.

**Graph of MobileNet:**

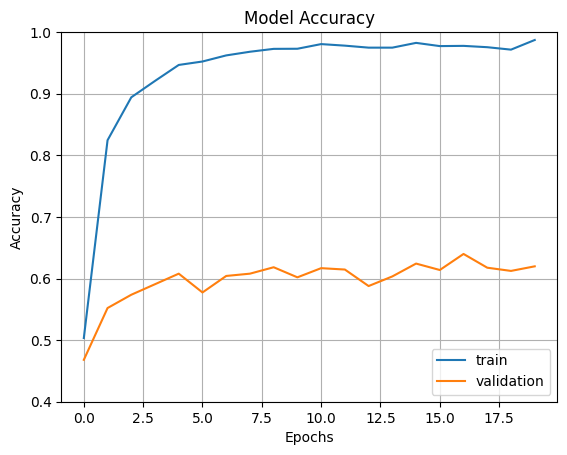


Figure 15: MobileNetV3 model performance during the training and validation process

Figure 15:MobileNet model performance during the training and validation process shows the model accuracy over 20 epochs for both the training and validation datasets. The training accuracy (blue line) increases rapidly, reaching around 0.95 within the first few epochs and maintaining that level with minor fluctuations. The validation accuracy (orange line) improves more gradually, fluctuating between 0.5 and 0.65 throughout the training period. The noticeable difference between the high training accuracy and the lower, more unstable validation accuracy indicates that the model is likely overfitting, performing well on the training data but not generalising effectively to the validation data.

**Graph of VGG16:**

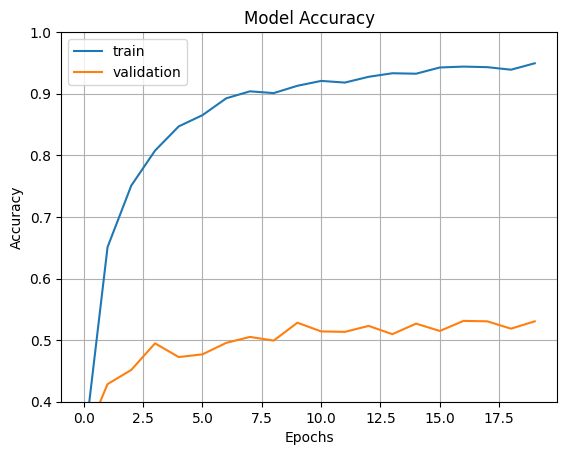


Figure 16: VGG16 model performance during the training and validation process

Figure 16:VGG16 model performance during the training and validation process shows a model's accuracy over 20 epochs on both training and validation datasets. The blue line reflects training accuracy, which progressively climbs and exceeds 90% by the tenth epoch, suggesting that the model is learning effectively from the training data. In comparison, the orange line depicts the validation accuracy, which begins at 40%, rises slightly above 50%, and then varies around this number. The difference in training and validation accuracy indicates that the model is overfitting to the training data, as it performs substantially better on training data than on validation data.

**Conclusion:**

This research aimed to compare how well several advanced pre-trained Convolutional Neural Networks (CNNs)—MobileNetV3, VGG16, EfficientNetB7, and ResNet-50—performed on the Indian Medicinal Leaves Image Dataset using transfer learning. Each model's performance was evaluated based on training accuracy, validation accuracy, loss, and confusion matrix. The experiments were conducted in a Jupyter Notebook using Python version 3.12.2 on a MacBook Air with an Apple M1 chip.

EfficientNetB7 showed the highest training accuracy, steadily improving throughout training and reaching an impressive 98.88% by the end. However, its validation accuracy varied more, peaking at 66.22% and slightly decreasing later, indicating potential overfitting. The loss values followed a similar pattern, with training loss decreasing significantly but validation loss fluctuating.

ResNet-50 also performed well in training, achieving 96.77% accuracy by the end. Its validation accuracy, although lower than EfficientNetB7, remained more stable around 57-60%, suggesting better generalization. Training loss decreased consistently, but validation loss showed a slight increase, suggesting mild overfitting.

VGG16 achieved a high training accuracy of 94.71% but struggled with validation accuracy, fluctuating around 51-53%. Increasing validation loss indicated overfitting despite its simple and deep architecture.

MobileNetV3 demonstrated balanced performance with training accuracy matching EfficientNetB7 at 98.88% and stable validation accuracy around 61-63%. Its loss trends were more consistent, suggesting effective learning and generalization.

In summary, EfficientNetB7 and ResNet-50 excelled in training accuracy, while MobileNetV3 showed balanced performance with effective generalization. VGG16, despite good training accuracy, faced challenges in generalizing to validation data due to overfitting. These findings underscore the importance of selecting the right pre-trained model based on specific needs such as accuracy, efficiency, and generalization capabilities, offering valuable insights for practical applications in agriculture, medicine, and environmental monitoring using CNNs.

**Reference:**

For a comprehensive guide and additional resources on completing this project, please refer to the YouTube playlist:

[Complete Deep Learning and Image Processing Project](https://youtube.com/playlist?list=PLKnIA16_RmvYuZauWaPlRTC54KxSNLtNn&si=s9MAnxWzlAGmaq3P).

**Research Papers:**

1. [Geerthana, R., P. Nandhini, and R. Suriyakala. "Medicinal Plant Identification Using Deep Learning."](https://www.semanticscholar.org/paper/Medicinal-Plant-Identification-Using-Deep-Learning-Geerthana-Nandhini/23b4104f71d90458d286ece5b69644eba7ada4fa?p2df)*[International Research Journal on Advanced Science Hub](https://www.semanticscholar.org/paper/Medicinal-Plant-Identification-Using-Deep-Learning-Geerthana-Nandhini/23b4104f71d90458d286ece5b69644eba7ada4fa?p2df)*[3 (2021): 48-53.](https://www.semanticscholar.org/paper/Medicinal-Plant-Identification-Using-Deep-Learning-Geerthana-Nandhini/23b4104f71d90458d286ece5b69644eba7ada4fa?p2df)
2. [Prasvita, Desta Sandya, and Yeni Herdiyeni. "MedLeaf: mobile application for medicinal plant identification based on leaf image."](https://www.researchgate.net/profile/Yeni-Herdiyeni/publication/237441676_MedLeaf_Mobile_Application_For_Medicinal_Plant_Identification_Based_on_Leaf_Image/links/0046351bb5c9fcfa1d000000/MedLeaf-Mobile-Application-For-Medicinal-Plant-Identification-Based-on-Leaf-Image.pdf)*[International Journal on Advanced Science, Engineering and Information Technology](https://www.researchgate.net/profile/Yeni-Herdiyeni/publication/237441676_MedLeaf_Mobile_Application_For_Medicinal_Plant_Identification_Based_on_Leaf_Image/links/0046351bb5c9fcfa1d000000/MedLeaf-Mobile-Application-For-Medicinal-Plant-Identification-Based-on-Leaf-Image.pdf)*[3.2 (2013): 5-8.](https://www.researchgate.net/profile/Yeni-Herdiyeni/publication/237441676_MedLeaf_Mobile_Application_For_Medicinal_Plant_Identification_Based_on_Leaf_Image/links/0046351bb5c9fcfa1d000000/MedLeaf-Mobile-Application-For-Medicinal-Plant-Identification-Based-on-Leaf-Image.pdf)
3. [Kavitha, S., et al. "Medicinal Plant Identification in Real-Time Using Deep Learning Model."](https://link.springer.com/article/10.1007/s42979-023-02398-5)*[SN Computer Science](https://link.springer.com/article/10.1007/s42979-023-02398-5)*[5.1 (2023): 73.](https://link.springer.com/article/10.1007/s42979-023-02398-5)
4. [Azadnia, Rahim, et al. "An AI-based approach for medicinal plant identification using deep CNN based on global average pooling." *Agronomy* 12.11 (2022): 2723.](https://www.mdpi.com/2073-4395/12/11/2723)
5. [Sainin, Mohd Shamrie, Taqiyah Khadijah Ghazali, and Rayner Alfred. "Malaysian medicinal plant leaf shape identification and classification." (2014): 1-6.](https://repo.uum.edu.my/id/eprint/12429/1/rie%281%29.pdf)
6. [Rao, R. Upendar, et al. "Identification of medicinal plants using deep learning." *Int J Res Appl Sci Eng Technol* 10 (2022): 306-22.](https://d1wqtxts1xzle7.cloudfront.net/83736174/Identification_of_Medicinal_Plants_using_Deep_Learning-libre.pdf?1649644492=&response-content-disposition=inline%3B+filename%3DIdentification_of_Medicinal_Plants_using.pdf&Expires=1720166078&Signature=P~Xndl94O47-xkGraD9Bze4MBTMC5PDF~0zyFa919EHpXsMj84lnLprbHpodXVll4ISC7QECzEs2EKyLXKo8pM1pNWr0T8oAN3Y55JAgBKoy0hyLKsTv-vIz9I~5yWQ3THS~NyNlI~IAqhG-ETXziIl6rBhnL47Dbzqd5U5RWhTGszlltx7k9pHgGbVf3wW2Vt1PXkn57DeYHi4y8jLG4p2udngVcxwYFuN-bpIgJi4gpWIFhtgm3STyOYGewFdsx-wf4Ndxdqg-jL4-5yGcskHMmdkqCsGaOhIzW4kYtLBG74bdimWsP8eX0jA4cCIHzto0gS~zNTn3C6HpsvXBcg__&Key-Pair-Id=APKAJLOHF5GGSLRBV4ZA)