Frontiers in Health Informatics
ISSN-Online: 2676-7104
www.healthinformaticsjournal.com
2024; Vol 13: Issue 2
Open Access

# "Empirical Analysis of Transformer Models and Pretrained Convolutional Neural Networks for Medicinal Plant Identification and Classification"

[1]Mrs. Sheetal S. Patil*, [2]Dr. Suhas H. Patil, [3]Dr. Avinash M. Pawar, [4]Mrs. Gauri R. Rao, [5]Dr. Rohini B. Jadhav, [6]Dharmesh Dhabliya

[1]Computer Engineering Department, Bharati Vidyapeeth Deemed to be University College of Engineering Pune sspatil@bvucoep.edu.in

[2]Computer Engineering Department, Bharati Vidyapeeth Deemed to be University college of Engineering Pune shpatil@bvucoep.edu.in

[3]Bharati Vidyapeeth's College of Engineering for Women, Pune avinash.m.pawar@bharatividyapeeth.edu

[4]Computer Engineering Department, Bharati Vidyapeeth Deemed to be University college of Engineering Pune grrao@bvucoep.edu.in

[5]Department of Information Technology Department, Bharati Vidyapeeth Deemed to be University College of Engineering Pune rbjadhav@bvucoep.edu.in

[6]Professor Vishwakarma Institute of Information Technology, dharmesh.dhabliya@viit.ac.in

*Corresponding Author mail: sspatil@bvucoep.edu.in

**Article Info**

***Article type:***

*Research*

***Article History:***

*Received: 2024-04-06*

*Revised: 2024-05-24*

*Accepted: 2024-06-25*

***Keywords*:**

Computer Vision, Pretrained CNN, Transformer, EfficientNetB7, VIT, Medicinal plant classification.

**A B S T R A C T**

This paper explores the classification of medicinal plants using computer vision, specifically focusing on the effectiveness of pretrained models and vision transformers. The study aims to compare two classification approaches: transfer learning using pretrained models and the application of vision transformers, which leverage self-attention mechanisms. By utilizing a dataset comprising leaf images of 10 distinct medicinal plant species, we conducted an empirical analysis with an 80:20 split ratio, the data was separated into training and testing sets. Preprocessing techniques, including class balance and data augmentation (small adjustments to data like (cropping, adding noise) were used to optimize the dataset.

The EfficientNetB7, a pretrained Convolutional Neural Network (CNN), was employed, achieving a Test Accuracy: 98.25% with learning rate 1e-5. In contrast, the VIT (Vision Transformer) model, a vision transformer-based approach, outperformed EfficientNetB7(Pretrained CNN) a test accuracy of 98.80 % with learning rate 1e-4.Despite its higher accuracy, the VIT model required greater computational resources and longer training times.

## 1. INTRODUCTION:

India is home to approximately 8,000 plant species with recognized medicinal properties. According to figures from the World Health Organization, traditional medicine provides basic treatment for 80% of the people in low-income nations (WHO)[1]. The therapeutic use of plant extracts represents the largest application of plant species globally, surpassing even their use as food sources. This highlights the crucial

role of plant species in drug development, particularly during the COVID-19 pandemic, where species like *Allium cepa*, *Allium sativum*, and *Citrus spp.* have been effective in managing the disease.[2]

The use of Ayurvedic medicine, which originates in India, exemplifies the longstanding tradition of utilizing medicinal plants. Approximately 90% of Ayurvedic remedies are composed of plant extracts. The origins of Ayurvedic medicine date back to 500 BC at the University of Banaras, [3]where the foundational texts, such as the ancient *Samhitas*, were composed. Ayurveda is recognized as one of the oldest medical systems, predating even Chinese medicine, and is considered a science of life with established principles and practices aimed at restoring and maintaining health. Historical records suggest that the great physician Dioscorides, author of *Materia Medica*, traveled extensively and may have drawn inspiration from Indian medicinal practices[4] .

India's rich heritage in traditional medicine and ethnomedicine is well-documented, yet much of this knowledge remains in non-digital formats, creating barriers to modern drug development. Traditional Indian medicines are often complex, multi-ingredient formulations based on experiential knowledge rather than a theoretical understanding of the active components[5].

The traditional manual identification process used by botanical experts relies on specific plant features, which serve as identification keys. These features are systematically and adaptively tested to identify different species. The process involves answering a sequence of questions about the plant's characteristics, such as color, shape, number of petals, and the presence of hairs or thorns. This method gradually narrows down the possibilities until the species is identified. However, this approach demands significant botanical expertise, making it inaccessible to most nature enthusiasts and the general public. As a result, traditional plant species identification remains challenging and, in many cases, nearly impossible for non-experts[6].

The identification of medicinal plant species is a critical but complex task that poses challenges for both the general public and professionals. While traditional methods of plant identification offer reliable results, they are often inaccessible due to the expertise required. The digitization of traditional knowledge and the development of user-friendly identification tools are essential to overcome these barriers and support the continued exploration and utilization of medicinal plant species in drug development and healthcare.

Medicinal plants have been a cornerstone of traditional medicine for centuries, with many species still unknown to modern science. Accurate identification and classification of these plants are crucial for conservation, research, and healthcare applications. However, traditional methods rely on expert botanists, it is error-prone and time-consuming[6].

This research paper focuses on advances in computer vision, Recent developments in computer vision[7] [8]have provided powerful tools for automating the process of classification of medicinal plant species. Pretrained CNN models, which leverage transfer learning[9], have shown to be especially effective for image classification tasks., enabling high-performance results with reduced training times. In addition, transformers, a newer class of models based on self-attention mechanisms, have emerged as promising alternatives for image classification.

Both approaches were evaluated using various performance metrics to assess their effectiveness in classifying a 10 classes of medicinal plant species from dataset having Indian medicinal plants (Patil, Sheetal (2024)), [10].

## 2. RELATED WORK:

Kan et al. [11]proposed an automated classification approach to address the limitations of conventional algorithms in the identification of medicinal plants. The procedure starts with preprocessing images of leaves of medicinal plants, then extracts ten shape features (SF) and 5 texture features (TF). The leaves of the medicinal plants are then identified by using support vector machine (SVM) classifier. An average

accuracy rate achieved by the classification model was 93.3% when evaluated on images of twelve distinct leaves from medicinal plants.

Alimboyong et al.[12] proposed a computer vision-based method for recognizing Ayurvedic medicinal plant species found in the Western Ghats of India. The method is taking leaf image data and extracting features by combining Histogram of Oriented Gradients (HOG) and Speeded-Up Robust Features (SURF), followed by classification using a k-Nearest Neighbors (k-NN) classifier. The results of the study indicate that the method is sufficiently robust for the development of applications intended for real-world usage.

In this study[13] Ashutosh Kumar Singh et al have applied data augmentation to the Plant Village dataset images of apple, corn, potato, tomato, and rice plants, extracting deep features using a convolutional neural network (CNN). The methodology enables early intervention to prevent crop damage, thereby averting potential economic crises. Additionally, texture and color features were extracted using Histogram of Oriented Gradients (HoG), Gray-Level Co-occurrence Matrix (GLCM), and color moments. After combining these features to create hybrid features, a Bayesian-optimized support vector machine (SVM) classifier was used to classify the hybrid features. Then results measured in terms of precision, sensitivity, F-score, and accuracy.

The erosion of traditional knowledge, due to a lack of tools and technologies, represents a significant challenge. Bridging the gap between traditional and scientific methods is essential to preserve and effectively utilize this knowledge in modern applications. Traditional approaches lacked the ability to extract the target leaf automatically.

Pretrained neural networks and transformers have revolutionized automatic image classification by offering powerful, flexible, and efficient models. This research paper focuses on the empirical analysis of medicinal plant classification using EfficientNetB7[14] and Vision Transformer (ViT).[15]

## 3. METHODOLOGY:

### i. Dataset:

In this study, we employed a dataset[10] ,comprising 10 distinct classes of medicinal plants, each identified by their botanical and local names. The selected 10 classes for experimentation include *Mangifera indica* (Mango), *Mentha* (Mint), *Nerium oleander* (Oleander), *Piper betle* (Betel), *Ocimum tenuiflorum* (Tulsi), *Psidium guajava* (Guava), *Syzygium cumini* (Jamun), *Thespesia populnea* (Portia tree), *Santalum album* (Sandalwood), and *Trigonella foenum-graecum* (Fenugreek). This dataset, characterized by a substantial number of samples for each class, provided a robust foundation for our experimental analysis.

### ii. Data preprocessing steps:

**a) Step1. Addressing Class Imbalance and Quantifying Class Imbalance:** In many classification tasks, the distribution of classes within the dataset is often imbalanced, meaning that some classes are significantly more represented than others[16]. This class imbalance can have a detrimental effect on model performance, particularly in scenarios where the model becomes biased towards the majority class[17]. Such bias can lead to a high overall accuracy but poor performance on the minority classes, which are often of equal or greater importance in real-world applications. Addressing class imbalance is therefore crucial to ensure that the model can generalize well across all classes and does not underperform on underrepresented classes. To effectively address class imbalance, it is first necessary to quantify the extent of imbalance within the dataset. One approach to measure class imbalance is by calculating the class imbalance ratio. This ratio provides a numerical representation of how much more frequent the majority class is compared to the other classes. The formula for calculating this ratio is given by:

$$\boldsymbol{class\ imbalance\ ratio_c} = \frac{\max(n_k)}{n_c} \text{.....Equation 1}$$

Where:

**$\max(n_k)$**: This represents the maximum number of samples across all classes k. Essentially, it is the size of the largest class in the dataset.

**$n_c$** This indicates how many samples there are in class iii, which is the class you are currently evaluating.

**b) Step2. Data augmentation**: To increase how resilient the model is to changes in the image data; we implemented a comprehensive set of image augmentation techniques. These augmentations artificially increase the diversity of the training dataset, enhancing the model's capacity to generalize previously undiscovered data. The following transformations were applied:

**Contrast Adjustment (iaa.Sometimes(0.5, iaa.LinearContrast((0.8, 1.2))))**: Contrast adjustment was performed with a probability of 50%, where the contrast of the image was linearly scaled by a factor randomly selected between 0.8 and 1.2. Lowering the contrast factor reduces the differences between the lightest and darkest parts of the image, while increasing it enhances these differences. This augmentation helps the model become invariant to different lighting conditions.

**Horizontal Flipping (iaa.Fliplr(0.5))**: Horizontal flipping was applied with a 50% probability. This transformation reflects the image across its vertical axis, simulating variations in the object's orientation. This augmentation is particularly useful for datasets where the horizontal orientation of the object is not fixed or is irrelevant to the classification task.

**Rotation (iaa.Affine(rotate=(-20, 20)))**: A random rotation within the range of -20 to +20 degrees was applied to the images. This transformation accounts for variations in the angle at which objects are captured, making the model more robust to different perspectives and orientations.

**Brightness Adjustment (iaa.Multiply((0.8, 1.2)))**: The brightness of the images was modified by multiplying pixel values by a factor randomly chosen between 0.8 and 1.2. This adjustment simulates different lighting conditions, ensuring that the model performs consistently regardless of changes in image brightness.

**Gaussian Blur (iaa.GaussianBlur(sigma=(0.0, 1.0)))**: Gaussian blur was applied with a standard deviation (sigma) randomly selected between 0.0 and 1.0. This operation smooths the image by reducing noise and detail, helping the model to focus on the larger patterns rather than the finer details, which might be subject to noise.

**Scaling (iaa.Affine(scale=(0.8, 1.2)))**: Scaling was applied with a random factor between 0.8 and 1.2, either shrinking or enlarging the image while preserving its aspect ratio. This augmentation introduces variability in object size, making the model more adaptable to different scales of objects in the image.

**Additive Gaussian Noise (iaa.Sometimes(0.5, iaa.AdditiveGaussianNoise(scale=(0, 0.1 * 255))))**: Additive Gaussian noise was introduced with a probability of 50%. The noise intensity was scaled between 0 and 0.1 times the maximum pixel value (255), simulating various levels of sensor noise. This augmentation improves the model's ability to handle images with varying degrees of noise.

**Pixel Dropout (iaa.Sometimes(0.5, iaa.Dropout(p=(0, 0.1))))**: Pixel dropout was applied with a probability of 50%, where up to 10% of pixels were randomly set to zero. This augmentation forces the model to rely less on any single pixel, enhancing its robustness to missing or corrupted data.

**Salt-and-Pepper Noise (iaa.Sometimes(0.5, iaa.SaltAndPepper(0.05)))**: Salt-and-pepper noise was introduced with a probability of 50%, where 5% of pixels were randomly set to either the maximum (salt) or minimum (pepper) value. This noise simulates extreme cases of image degradation, preparing the model to handle similar issues in real-world data.

**c) Step3.: Image Quality Assurance: Removal of Corrupted Images**

In the process of preparing the dataset for training, It is crucial to verify that all images are of high quality and free from corruption.. Corrupted images can arise from incomplete downloads, file transfer errors, or

other anomalies, and their presence in the dataset can lead to errors during training and adversely affect model performance. Therefore, a methodical approach was employed to detect and remove such corrupted images.

Table 1 gives a summary of the total number of images per plant species after augmentation and Figure 1 presents sample images from the dataset.

*Table 1: Total number of images after augmentation per class*

| Sr. No | Sub-folder | Total No of images |
|---|---|---|
| 0 | Psidium Guajava (Guava) | 1992 |
| 1 | Ocimum Tenuiflorum (Tulsi) | 2015 |
| 2 | Nerium Oleander (Oleander) | 1717 |
| 3 | Trigonella Foenum-graecum (Fenugreek) | 1785 |
| 4 | Mangifera Indica (Mango) | 1932 |
| 5 | Mentha (Mint) | 1707 |
| 6 | Syzygium Cumini (Jamun) | 1855 |
| 7 | Thespesia populnea -(Portia tree) | 1772 |
| 8 | Piper Betle (Betel) | 1790 |
| 9 | Santalum Album (Sandalwood) | 1928 |


*Figure 1:random samples from dataset*[10]

**iii. Process for Empirical Analysis.**

We first trained the Vision Transformer (VIT) classifier model on a dataset comprising 18,472 images after augmentation across 10 classes, Subsequently, we evaluated the performance of the trained VIT model on a test dataset consisting of 1,245 images. The model was trained with a batch size of 32 and a learning rate learning rate of 1e-4 using the Adam optimizer. Training was conducted over 40 epochs, with early stopping employed to mitigate the risk of overfitting.

**A. Vision Transformer (ViT) Model:**

VIT: A model architecture that uses the transformer concept is called the Vision Transformer (ViT), originally designed for NLP tasks, to image classification[18]. Unlike traditional convolutional neural networks (CNNs), which use convolutions to extract local features, ViT treats an image as a sequence of patches, akin to tokens in a text sequence. This technique uses transformers' self-attention mechanism to capture global context, making it especially useful for large-scale image categorization applications.

The architecture of the Vision Transformer (ViT) can be broken down into several key components:

The Vision Transformer model consists of several key components:[15]

**a.Patch Embedding Layer**

**b.Transformer Encoder**

**c.Classification Head**

**a.Patch Embedding Layer:**

The supplied image is split up into non-overlapping, fixed-size patches. After that, each patch is linearly embedded into a vector, which is used as the transformer's input.

**Input Image**: An image of size H×W×C (e.g.224*224*3 for RGB images)

**Patch Size**: P×P(e.g., 16x16 pixels).

**Number of Patches**: The image is split up into $\frac{H}{P} \times \frac{W}{P}$ patches resulting in a total of N patches.

**Flattening and Linear Projection**: Each P×P×C patch is flattened into a vector of size $P^2 \times C$ and projected into an embedding space of dimension D (embedding dimension).

The sequence of patch embeddings is combined with positional encodings to maintain spatial information:

$z0 = [E1; E2; \dots; EN] + Epos$ ..... Equation 2

where $Ei$ is the embedding of patch i, and Epos is the positional encoding.

**b.Transformer Encoder**

The sequence of patch embeddings is processed by a stack of L identical transformer encoder layers. Each layer comprises:

**Multi-Head Self-Attention (MHSA)**: The model may simultaneously focus on several areas of the image due to the self-attention mechanism. The attention is computed as:

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{D_K}}\right)V \text{.....Equation 3}$$

where the queries, keys, and values are denoted by Q, K, and V, respectively, and $D_K$ is the dimensionality of the keys.

Feed-Forward Network (FFN): A position-wise feed-forward network made up of two linear layers separated by a ReLU activation function is included in each transformer layer. The FFN enhances the model's capacity to learn complex representations.

**Layer Normalization and Residual Connections**: To sustain training and enhance gradient flow, layer normalization and residual connections come after each MHSA and FFN.

**c.Classification Head**

After passing through the transformer encoder layers, the output corresponding to a special [CLS] token (classification token) is used for classification:

**[CLS] Token**: The patch embedding sequence is prepended with a unique token. The final representation of this token is used to predict the class label.

**Fully Connected Layer**: To generate class probabilities, the [CLS] token's output is fed into a dense layer with a softmax activation function.

The ViT-Base/16 model has the following key configurations:

Patch Size: 16x16 pixels

Input Resolution: 224x224 pixels

Number of Transformer Layers: 12

Embedding Dimension: 768

Number of Attention Heads: 12

Feed-Forward Network (FFN) Dimension: 3072

Number of Parameters: Approximately 86 million

Pre-training Dataset: ImageNet-21k

Fine-tuning Dataset: ImageNet-1k (optional)

Figure 2 below provides a detailed illustration of the transformer architecture.

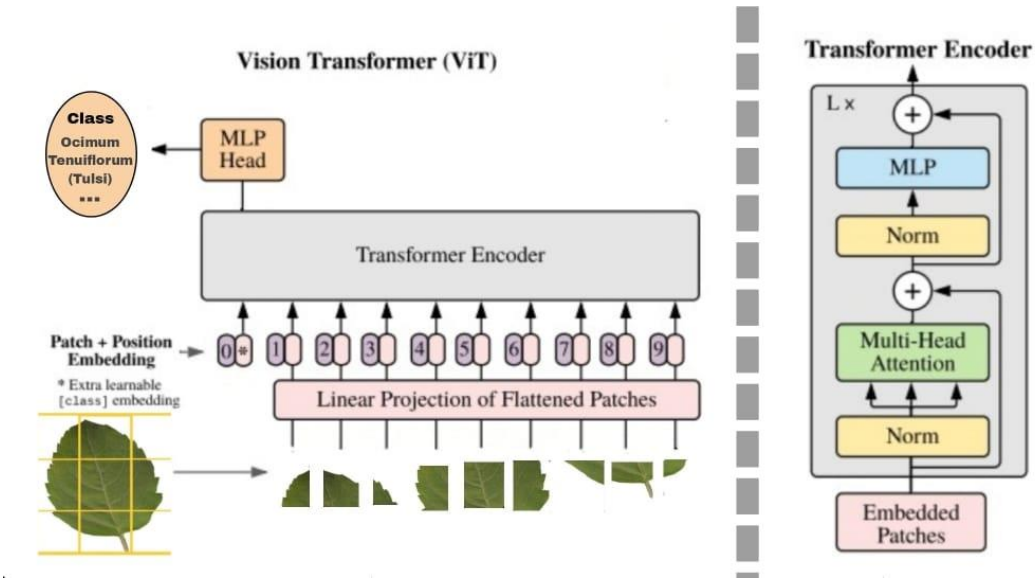



*Figure 2:image is divided into patches and position embeddings are done, resulting vector sequence is given to transformer encoder* [19]

**B. EfficientNetB7:**

EfficientNet-B7 is an advanced model within the EfficientNet family[20], distinguished by its exceptional performance and efficiency in image classification tasks. It extends the architecture of EfficientNet-B0 through a process of compound scaling, resulting in a more powerful and accurate model. The architecture of EfficientNet-B7 is designed to maximize performance while optimizing computational resources. For empirical analysis Model was trained using optimizer Adam (learning rate=0.00001) for 150 epochs.

**Architectural Components:**

1. **Baseline Network**: EfficientNet-B7 builds on EfficientNet-B0, which serves as the baseline network. EfficientNet-B0 employs a combination of depthwise separable convolutions and Mobile Inverted Bottleneck Convolutions (MBConv) to achieve a compact and efficient network structure.
2. **Compound Scaling**: The key innovation of EfficientNet-B7 is its use of compound scaling. The depth, width, and input resolution of the network are all uniformly scaled using this method. The scaling process follows a balanced approach to enhance performance without disproportionally increasing computational complexity. EfficientNet-B7 is scaled from EfficientNet-B0 by using specific scaling factors for depth, width, and resolution:

   **Depth**: The number of layers in the network is increased.

   **Width**: The number of channels per layer is expanded.

   **Resolution**: The input image resolution is increased.

   The formula for compound scaling is:

   Scaling Factor

   $New\ Dimension = Baseline\ Dimension\ Scaling\ Factor$ .....*Equation 4*
3. **Mobile Inverted Bottleneck Convolutions (MBConv)**: EfficientNet-B7 incorporates MBConv blocks, which are a variant of the Inverted Residual Block from MobileNetV2. Each MBConv block consists of:

   **Depth wise Separable Convolutions**: To reduce computational complexity, these convolutions divide the filtering process into depth wise and pointwise convolutions.

   **Expansion Layer**: Prior to applying depth wise separable convolutions, an expansion layer increases the number of channels.

   **Residual Connections**: Shortcut connections are used to facilitate the flow of gradients and improve training efficiency.
4. **Swish Activation Function**: The Swish function of activation is described as

   Swish(x)=x·sigmoid(x), ..... Equation 5

   is used throughout the EfficientNet-B7 architecture. Swish has been shown to provide better performance compared to the traditional ReLU activation function by enabling smoother gradient flow and enhancing the model's representational capacity.
5. **Network Architecture**: EfficientNet-B7 consists of a series of MBConv blocks arranged in stages. The architecture can be summarized as follows:

   **Stem Layer**: The input image is first processed by a stem convolution layer that reduces the spatial dimensions and extracts initial features.

   **MBConv Stages**: The network is divided into several stages, each comprising multiple MBConv blocks. The number of blocks and channels increases progressively across stages.

**Global Average Pooling**: After the final MBConv stage, feature maps are combined using global average pooling to create a fixed-size vector.

**Fully Connected Layer**: The pooled feature vector is fed into a fully connected layer, followed by a softmax activation function to produce class probabilities.

6. **Scaling Factors**: EfficientNet-B7 employs scaling factors to increase the depth, width, and resolution of the network. Specifically, EfficientNet-B7 has:

**Depth**: 66 layers

**Width**: 32 channels per layer

**Resolution**: 600 x 600 pixels input size., Figure 3 gives details of general architecture of the EfficientNet

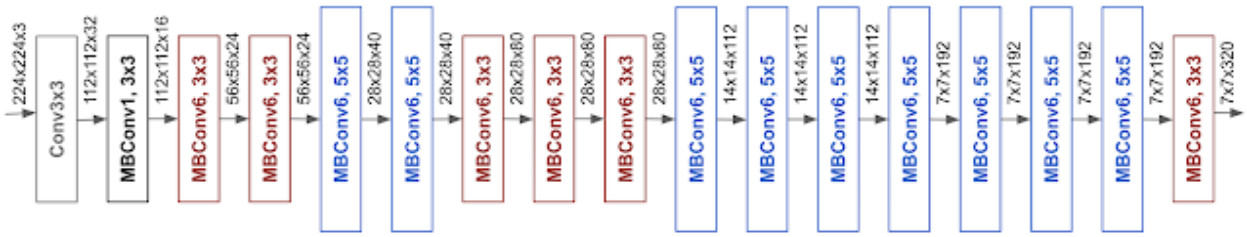



*Figure 3 : The architecture of EfficientNet.* [14]

**C. Experimental Setup**

**i) Hardware Configuration:**

The experiments were conducted using Google Colab Pro, which provides access to a Tesla T4 GPU. The specific hardware configuration is as follows:

**GPU**: NVIDIA Tesla T4

**Tensor Cores**: 320

**CUDA Cores**: 2560

**Memory**: 16 GB GDDR6

**Compute Capability**: 7.5

The Tesla T4 GPU is optimized for deep learning workloads, providing accelerated performance for training and inference of neural network models.

**ii) Software Environment:**

The software environment for the experiments was set up using the following tools and libraries:

**Operating System**: Ubuntu 18.04 LTS (underlying system on Google Colab)

**Programming Language**: Python 3.9

**Deep Learning Framework**: PyTorch 1.13.1

**Transformers Library**: Hugging Face Transformers 4.26.0

**CUDA Toolkit**: CUDA 11.2

**cuDNN Version**: cuDNN 8.1

**Additional Libraries**: NumPy, Pandas, Matplotlib, Scikit-learn, and OpenCV for data preprocessing and evaluation.

**iii) Vision Transformer (ViT) Model Configuration**

Model Variant: ViT-B/16

Patch Size: 16x16 pixels

Input Image Size: 224x224 pixels

Number of Layers: 12 Transformer encoder layers

Number of Attention Heads: 12

MLP Dimension: 3072

Hidden Dimension: 768

Number of Parameters: Approximately 85 million

**iv) Training and optimization settings were used for VIT:**

Batch Size: 32

Number of Epochs: 40

Optimizer: AdamW (Adaptive Moment Estimation with Weight Decay)

Learning Rate: 1e-4 with a cosine annealing scheduler

Weight Decay: 0.01

Loss Function: Cross-Entropy Loss

Evaluation Metrics: Precision, Accuracy, F1-Score, and Recall

**v) EfficientNet-B7 Model Configuration**

Model Variant: EfficientNet-B7

Input Image Size: 600x600 pixels

Number of Layers: 66 layers (includes convolutional layers, batch normalization layers, and swish activation functions)

Compound Scaling Factors:

Depth: 2.0x (compared to EfficientNet-B0)

Width: 2.5x (compared to EfficientNet-B0)

Resolution: 2.0x (compared to EfficientNet-B0)

Number of Parameters: Approximately 66 million

Dropout Rate: 0.5

Activation Function: Swish

SE Ratio: 0.25 (Squeeze-and-Excitation blocks used in certain layers)

**vi)Training and optimization settings were used for EfficientNet-B7:**

Batch Size: 32

Epochs: 150 (with early stopping based on validation loss)

Optimizer: AdamW with a learning rate of 1e-5 and weight decay of 0.01

Loss Function: Cross-Entropy Loss

Evaluation Metrics: Precision, Accuracy, F1-Score, and Recall

## RESULTS:

**Precision:** For a given plant (e.g., Ocimum Tenuiflorum (Tulsi)), if the model predicts it as Ocimum Tenuiflorum (Tulsi), precision tells you how many of those predictions are correct. **Recall:** For Ocimum Tenuiflorum (Tulsi), recall indicates how many of the actual Ocimum Tenuiflorum (Tulsi)instances were correctly identified by the model. **F1-Score**: is a statistic that evaluates the model's performance by combining precision and recall. For Ocimum Tenuiflorum (Tulsi), the F1 score reflects the model's balance between precision and recall in identifying Ocimum Tenuiflorum (Tulsi) instances.

**Support:** The number of instances of Ocimum Tenuiflorum (Tulsi) in the dataset, which could affect how confident you are in the precision, recall, and F1-score. table 2 below give the details of recall, F1 score , precision and support values for VIT similarly table 3 below gives details of evaluation metrics for EfficientNet-B7.

*Table 2:Evaluation metrics for VIT*

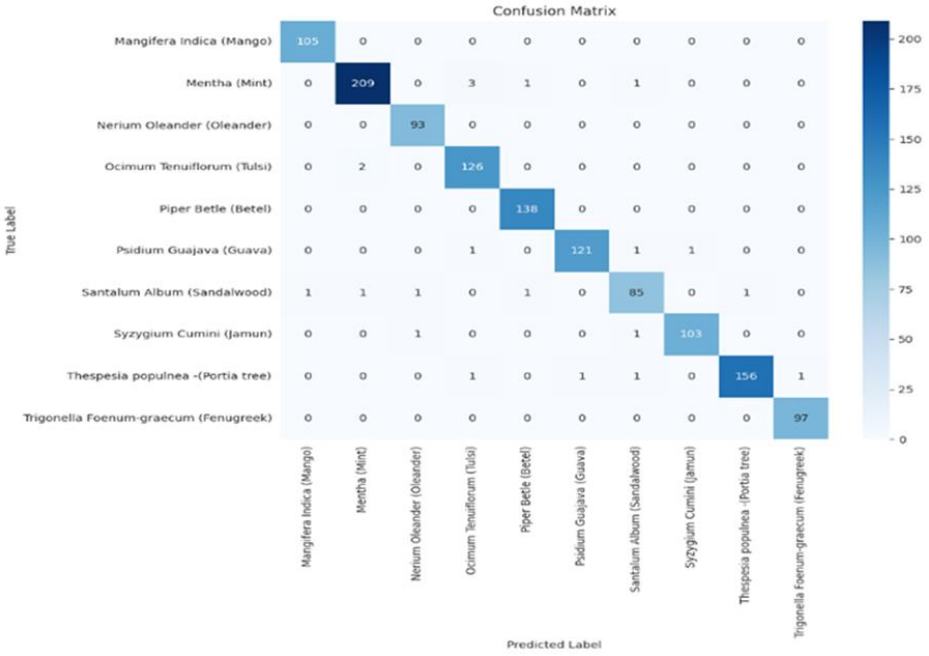
| Class | Precision | Recall | F1-Score | Support |
|---|---|---|---|---|
| Mangifera Indica (Mango) | 1 | 1 | 1 | 99 |
| Mentha (Mint) | 0.98 | 0.98 | 0.98 | 211 |
| Nerium Oleander (Oleander) | 1 | 1 | 1 | 93 |
| Ocimum Tenuiflorum (Tulsi) | 0.98 | 0.98 | 0.98 | 128 |
| Piper Betle (Betel) | 1 | 1 | 1 | 138 |
| Psidium Guajava (Guava) | 0.98 | 0.98 | 0.98 | 124 |
| Santalum Album (Sandalwood) | 0.95 | 0.97 | 0.96 | 90 |
| Syzygium Cumini (Jamun) | 1 | 1 | 1 | 105 |
| Thespesia populnea (Portia tree) | 0.99 | 0.99 | 0.99 | 160 |
| Trigonella Foenum-graecum (Fenugreek) | 0.99 | 1 | 0.99 | 97 |
| **Overall Metrics** | | | | |
| Accuracy | | | 0.99 | 1245 |
| Macro Avg | 0.99 | 0.99 | 0.99 | 1245 |
| Weighted Avg | 0.99 | 0.99 | 0.99 | 1245 |

*Table 3:Evaluation Metrics: for EfficientNet-B7*

| Class | Precision | Recall | F1-Score | Support |
|---|---|---|---|---|
| Mangifera Indica (Mango) | 0.99 | 1 | 1 | 105 |
| Mentha (Mint) | 0.99 | 0.98 | 0.98 | 214 |
| Nerium Oleander (Oleander) | 0.98 | 1 | 0.99 | 93 |
| Ocimum Tenuiflorum (Tulsi) | 0.96 | 0.98 | 0.97 | 128 |
| Piper Betle (Betel) | 0.99 | 1 | 0.99 | 138 |
| Psidium Guajava (Guava) | 0.99 | 0.98 | 0.98 | 124 |
| Santalum Album (Sandalwood) | 0.96 | 0.94 | 0.95 | 90 |
| Syzygium Cumini (Jamun) | 0.99 | 0.98 | 0.99 | 105 |
| Thespesia populnea (Portia tree) | 0.99 | 0.97 | 0.98 | 160 |
| Trigonella Foenum-graecum (Fenugreek) | 0.99 | 1 | 0.99 | 97 |
| **Overall Metrics** | | | | |

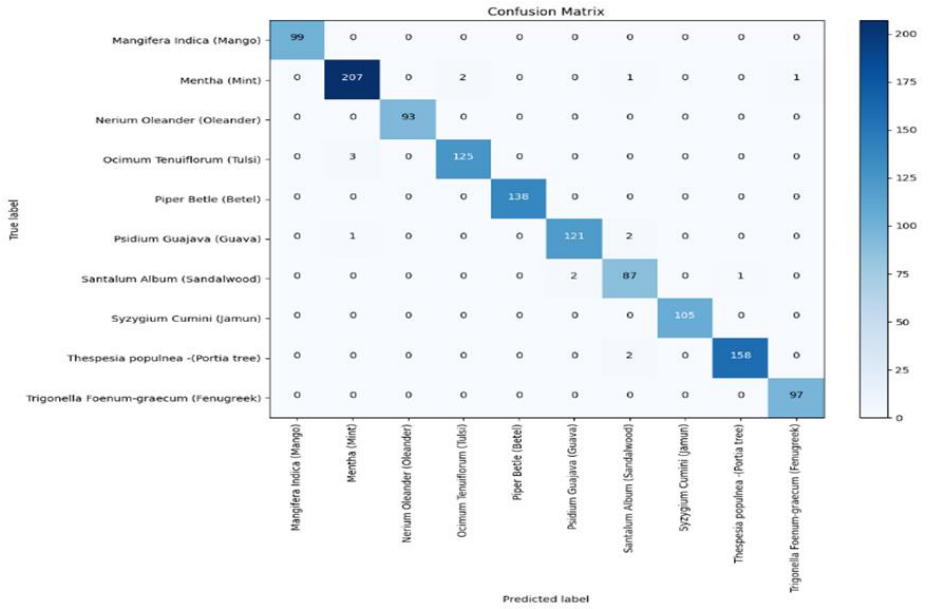
| | | | | |
|---|---|---|---|---|
| Accuracy | | | 0.98 | 1245 |
| Macro Avg | 0.98 | 0.98 | 0.98 | 1245 |
| Weighted Avg | 0.98 | 0.98 | 0.98 | 1245 |

The confusion matrices for the ViT and EfficientNetB7 models are shown in Figures 4 and 5, respectively.




**EfficientNet-B7: Test Loss: 0.06871 and Test Accuracy 98.25%**

**Figure 4: Confusion Matrix for EfficientNet-B7**




**Vision Transformer (VIT) Test Loss: 0.04973 and Test Accuracy: 98.80 %**

**Figure 5: Confusion Matrix for Vision Transformer (VIT)**

## CONCLUSION:

This study compares the performance of two advanced deep learning models, Vision Transformer (VIT) and EfficientNet-B7, for classifying 10 medicinal plant species.

The VIT model achieved an impressive overall accuracy of 99% on 1,245 test samples, with perfect precision, recall, and F1-scores (1.00) for four classes, and slightly lower scores (0.96) for *Santalum Album* (Sandalwood). The macro and weighted averages for all metrics achieved a value of 0.99 for the model.

EfficientNet-B7 also performed well, with an accuracy of 98% on 1245 test samples, the model shows high overall correctness in classifying medicinal plants. The macro average precision of 0.98 indicates that the model effectively identifies plants with minimal false positives, while the recall of 0.98 reflects its ability to detect nearly all instances across classes. The F1-score, also at 0.98, demonstrates a well-balanced performance between precision and recall, ensuring consistent results across all plant categories.

Although both models showed high accuracy and robust classification capabilities, VIT slightly outperformed EfficientNet-B7, particularly in consistency and overall accuracy. This indicates the potential of transformer-based architectures like VIT in image classification, though EfficientNet-B7 remains a strong and reliable alternative.

In conclusion, VIT offers superior performance, but both models are effective, and the choice between them should consider specific application needs and computational resources.

## REFERENCES:

[1] WHO, "General Guidelines for Methodologies on Research and Evaluation of Traditional Medicine World Health Organization," pp. 1–73, 2000, [Online]. Available: http://apps.who.int/iris/bitstream/10665/66783/1/WHO_EDM_TRM_2000.1.pdf (Accessed 09.09.2016)

[2] S. S. Patil, S. H. Patil, A. M. Pawar, N. S. Patil, and G. R. Rao, "Automatic Classification of Medicinal Plants Using State-Of-The-Art Pre-Trained Neural Networks," *Journal of Advanced Zoology*, vol. 43, no. 1, 2022.

[3] S. S. Sanjay Srivastava, "Herbal Cures Practised By Rural Populace In Varanasi Region Of Eastern U.P.(India)," *IOSR J Pharm Biol Sci*, vol. 6, no. 1, pp. 1–5, 2013, doi: 10.9790/3008-0610105.

[4] N. Botanicae, H. Agrobotanici, J. Janick, and J. Stolarczyk, "Ancient Greek Illustrated Dioscoridean Herbals: Origins and Impact of the Juliana Anicia Codex and the Codex Neopolitanus", [Online]. Available: www.notulaebotanicae.ro

[5] R. K. Upadhyay, "Tulsi: A holy plant with high medicinal and therapeutic value," *International Journal of Green Pharmacy*, vol. 11, no. 1, pp. S1–S12, 2017.

[6] C. S. Campbell and C. Jeffrey, "An Introduction to Plant Taxonomy.," *Bulletin of the Torrey Botanical Club*, vol. 110, no. 2, p. 231, 1983, doi: 10.2307/2996350.

[7] M. Naveenkumar and V. Ayyasamy, "OpenCV for Computer Vision Applications," *Proceedings of National Conference on Big Data and Cloud Computing (NCBDC'15)*, no. March 2015, pp. 52–56, 2016, [Online]. Available: https://www.researchgate.net/publication/301590571_OpenCV_for_Computer_Vision_Applications

[8] Y. L. B, F. Yu, M. Kumar, and K. Reddy, *Computer Vision – ECCV 2020*, vol. 12350. 2020. [Online]. Available: http://dx.doi.org/10.1007/978-3-030-58558-7_8%0Ahttps://link.springer.com/10.1007/978-3-030-58558-7

[9] A. Beikmohammadi and K. Faez, "Leaf Classification for Plant Recognition with Deep Transfer Learning," *Proceedings - 2018 4th Iranian Conference of Signal Processing and Intelligent Systems, ICSPIS 2018*, no. December, pp. 21–26, 2018, doi: 10.1109/ICSPIS.2018.8700547.

[10] sheetal patil, "Medicinal Plant Leaf Dataset with name table(mostly found in Paschim Maharashtra.)," mendeley data.

[11] H. X. , J. L. & Z. F. L. Kan, "Classification of medicinal plant leaf image based on multi-feature extraction," 2017.

[12] C. R. Alimboyong and A. A. Hernandez, "An Improved Deep Neural Network for Classification of Plant Seedling Images," *Proceedings - 2019 IEEE 15th International Colloquium on Signal Processing and its Applications, CSPA 2019*, no. August, pp. 217–222, 2019, doi: 10.1109/CSPA.2019.8696009.

[13] A. K. Singh, S. Sreenivasu, U. S. B. K. Mahalaxmi, H. Sharma, D. D. Patil, and E. Asenso, "Hybrid Feature-Based Disease Detection in Plant Leaf Using Convolutional Neural Network, Bayesian Optimized SVM, and Random Forest Classifier," *J Food Qual*, vol. 2022, pp. 1–16, 2022, doi: 10.1155/2022/2845320.

[14] S. S. E. and Q. V. L. P. S. G. A. Mingxing Tan, "EfficientNet: Improving Accuracy and Efficiency through AutoML and Model Scaling," https://research.google/blog/efficientnet-improving-accuracy-and-efficiency-through-automl-and-model-scaling/.

[15] A. Dosovitskiy *et al.*, "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale," Oct. 2020, [Online]. Available: http://arxiv.org/abs/2010.11929

[16] M. Buda, A. Maki, and M. A. Mazurowski, "A systematic study of the class imbalance problem in convolutional neural networks," Oct. 2017, doi: 10.1016/j.neunet.2018.07.011.

[17] R. Shwartz-Ziv, M. Goldblum, Y. L. Li, C. B. Bruss, and A. G. Wilson, "Simplifying Neural Network Training Under Class Imbalance," Dec. 2023, [Online]. Available: http://arxiv.org/abs/2312.02517

[18] K. J. J. C. H. X. and J. P. Y. Huo, "Vision Transformer (ViT)-based Applications in Image Classification," in *2023 IEEE 9th Intl Conference on Big Data Security on Cloud (BigDataSecurity), IEEE Intl Conference on High Performance and Smart Computing, (HPSC) and IEEE Intl Conference on Intelligent Data and Security (IDS), New York, NY, USA, 2023,* BigDataSecurity-HPSC-IDS58521.2023.00033., 2023, pp. 135–140.

[19] A. Dosovitskiy *et al.*, "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale," Oct. 2020, [Online]. Available: http://arxiv.org/abs/2010.11929

[20] M. Tan and Q. V. Le, "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks," May 2019, [Online]. Available: http://arxiv.org/abs/1905.11946