AyurBotanica – Identification of Medicinal Plants using Deep Learning

A. Satyanaryana, K. S. J. Sneha

Computer Science Department,

Rajalakshmi Engineering College, Thandalam

***Abstract*—Medicinal plants have played an integral role in people's lives especially with the growing interest in Ayurveda and Natural remedies which bring about little to no side effects. Since ancient times, people have always turned to medicinal plants for therapeutic and medicinal values along with their potential health benefits. Identification of these medicinal plants may bring about a safe and effective way of using these plants to harbor the desired effect. Using deep learning by leveraging the power of transfer learning with EfficientNetV2 architecture, we make this process efficient and accurate. Specifically, Convolutional Neural Networks (CNNs) are utilized to detect these medicinal plants accurately. Unlike traditional methods, which may be laborious and prone to errors, our approach harnesses the efficiency and accuracy of deep learning.**

Keywords—Medicinal plants, Deep Learning, Transfer Learning, EfficientNetV2

I. INTRODUCTION

Medicinal plants constitute a huge part of the biodiversity on earth. They come in a variety of sizes, shapes and colours. But its benefits are underestimated. Traditional medicines work well with our bodies providing a complete cure without giving rise to another ailment. Their side effects are little to insignificant. They can pose a cure to a variety of diseases like respiratory diseases, heart ailments, dermatological diseases, reproductive problems and the list goes on. [1]

When we have such a rich biodiversity, utilising them to its maximum potential is in our hands. In order to do that we would have to identify the plant correctly.[2] This is the first step to utilising the plant's benefits. Unfortunately, there are a very few people who actually have good knowledge in regard to medical plants.

Due to the scarcity of knowledgeable professionals, it is hard to keep track of these plants. The existing methods include manual identification, where the professionals personally check the medicinal plants. But this process can take up a huge amount of time and is quite tiresome. Even if we could increase the number of professionals, it is quite difficult to spread them across various regions. Moreover, certain regions have local herbs and it's hard for everyone to know about the details of these herbs.[13-15]

Today, a lot of medical herbs and plants are becoming endangered. People are unaware of the plant's potential benefits and significance. So, identification of the medicinal plants becomes crucial.[4]

With the use of Convolutional Neural Network (CNN) we can identify these plants with better accuracy. In this project we are developing a model by leveraging the power of EfficientNetV2 and transfer learning to identify the medicinal plants with greater precision.[8]

Few examples of the medicinal plants are shown in Fig. 1

Fig 1. Pictures of some common Medicinal Plants

II. LITERATURE SURVEY

Identification of medicinal plants requires thorough knowledge of the plants and years of experience and expertise to give a good result. Using Deep learning, specifically Convolutional Neural networks is a great option as compared to the traditional manual identification of the plants done by the professionals.

If you take medicinal plants, there are a lot of features that can be extracted from them. Plants have distinct shapes, veins, colour, edges, stem etc making them unique. These features can be used by deep learning models to identify the plants. Feature extraction is one of the main processes of CNN.[3]

A. CONVOLUTIONAL NEURAL NETWORK

Neural Networks are the heart of deep learning. They mimic the functioning of the neurons in the human brain and hence the name. It consists of Node layers which can

be classified into input layer, few or more hidden layers and output layers. The input is fed in the input layer and the output which is either prediction or classification is derived from the output layer. In the hidden layer the processing occurs. There are weights and various mathematical operations which occur and the final output is just a series of mathematical operations [5]. CNN have changed the game of classification as it overcomes the manual laborious process of feature extraction. CNNs understand and extracts the features by itself.

The various layers in CNN are depicted in Fig. 2.

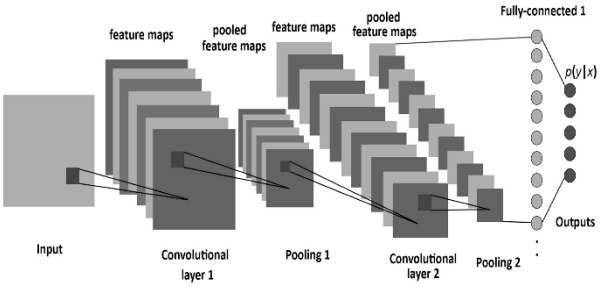

Fig 2. Layers in CNN

The general outline of the layers is as follows:

A.1. CONVOLUTIONAL LAYER

Convolutional layers are where a lot of computation occurs. It has the input layer, filter and feature maps.

The filters are applied across the input layer. The dot product of the pixels in the input image and the filter is taken and is fed to an output array. Then the filter moves by a stride. Stride is the number of pixels to move for the next iteration. And padding is done if necessary. Sometimes the filter might overlap the input image causing the resultant matrix to be of higher dimension or equal to the input image. To overcome this, we add zero padding. We can also increase the number of filters to increase the depth. These are few of the hyper parameters and by tuning these hyperparameters we can achieve a better result. The filter continues performing the dot product by moving until the entire image is swept by the kernel. Then reLu is applied to reduce the non-linearity in the data

The final output of the dot product of image and filter is what is called a feature map or activation map. It is depicted in Fig. 3.

CNN's aim is to convert every detail to a corresponding numerical value. Since machines work with numerical data, it would be easy for CNN to work on this data. [6,7]

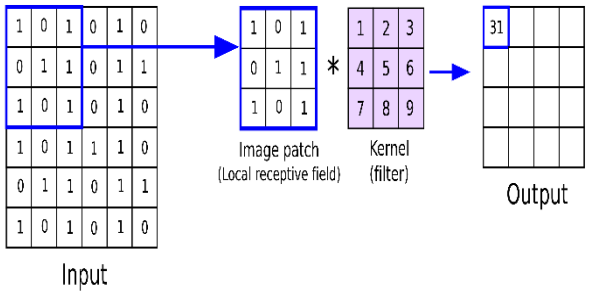

Fig 3. Convolutional Layer

A.2. POOLING LAYER

Often there might be unwanted features which might hinder the prediction or classification process. To reduce this, we go for dimensionality reduction. Pooling layer does this. It down samples the data and takes what's necessary. Pooling layer, much similar to the convolutional layer, uses filters and strides across the image until the entire image is swept by the kernel.

There are two types of pooling:

Max Pooling: It takes the maximum value of the set of values and places it in the output array.

Average Pooling: It takes the average value of the set of values and places it in the output array.[11]

Fig. 4 explains the Pooling layer's functionality

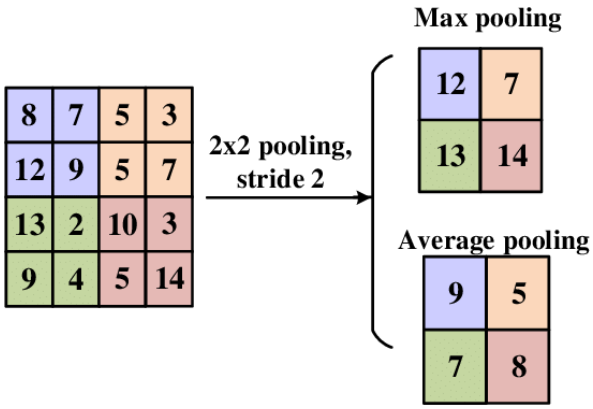

Fig 4. Pooling Layer

A.3. FULLY CONNECTED LAYER:

As the name suggests. All the nodes in the previous layer are connected to all the nodes in the output layer. It does the final classification from all the weights from the previous layer. Unlike convolutional layer and pooling layer, it uses SoftMax activation function to classify inputs. [12]

B. EFFICIENTNETV2

EfficientNetV2 is a deep learning model and a type of neural network architecture. It is proven to work faster and be more efficient than its previous model by being 6.8 times smaller. To optimise training speed and efficiency and parameter searching, it makes use of a combination of Neural Architecture Search and Scaling. New ops like Fused-MBConv. Helps enriching the process of searching the models from the search space.[9]

This model is pre-trained on ImageNet21k dataset with around 13 million images.

Being smaller and faster, it out beats various other traditional models.

C. TRANSFER LEARNING

Transfer Learning is a technique in which we reuse the knowledge from a pre trained model to improve the performance in a related task. These pre trained models are trained on large datasets. We will make use of these weights from the pretrained model and use it in our model. In this way we transfer the learning from the pre trained model and use it for our purpose [10]

III. PROPOSED METHODOLOGY

It focuses on the classification of different types of plant leaves using deep learning techniques. Development of the model architecture based on EfficientNetV2, which is pretrained with the ImageNet dataset and fine-tuned for plant leaf classification, forms an important part of this process. The dataset consists of images of various plant leaves, which are organized by class. The data preprocessing techniques include augmentation and others to increase model generalization. This model will be trained on the augmented data and performance on unseen test data will be shown. Furthermore, a Flask-based API is developed to provide real-time plant leaf-type prediction from a user-uploaded image, along with fetching information about the plant. This project is hence a robust and efficient solution for plant leaf recognition and information retrieval tasks.The architecture diagram is given in Fig.5

On uploading the image in the frontend, it goes to the backend and the model makes a prediction and the api sends back the prediction value to the user.

A. DATASET

This project will use a dataset of 2438 images of leaves from plants, categorized into three main sets: training, validation, and testing. The training set will have 1706 images, the validation set will have 366 images, and the testing set will have 366 images. These images span a wide range of 29 different classes, which define a number of plant species. Few plant species, among the rest, to be included in the dataset are aloe vera, arali, ashoka, bamboo, betel, camphor, coffee, drumstick, eucalyptus, ginger, guava, henna, hibiscus, jackfruit, jasmine, lantana, lemon, malabar nut, mango, mint, neem, papaya, pumpkin, rose, sapota, tamarind, tomato, tulsi, and turmeric leaves. The dataset is curated precisely

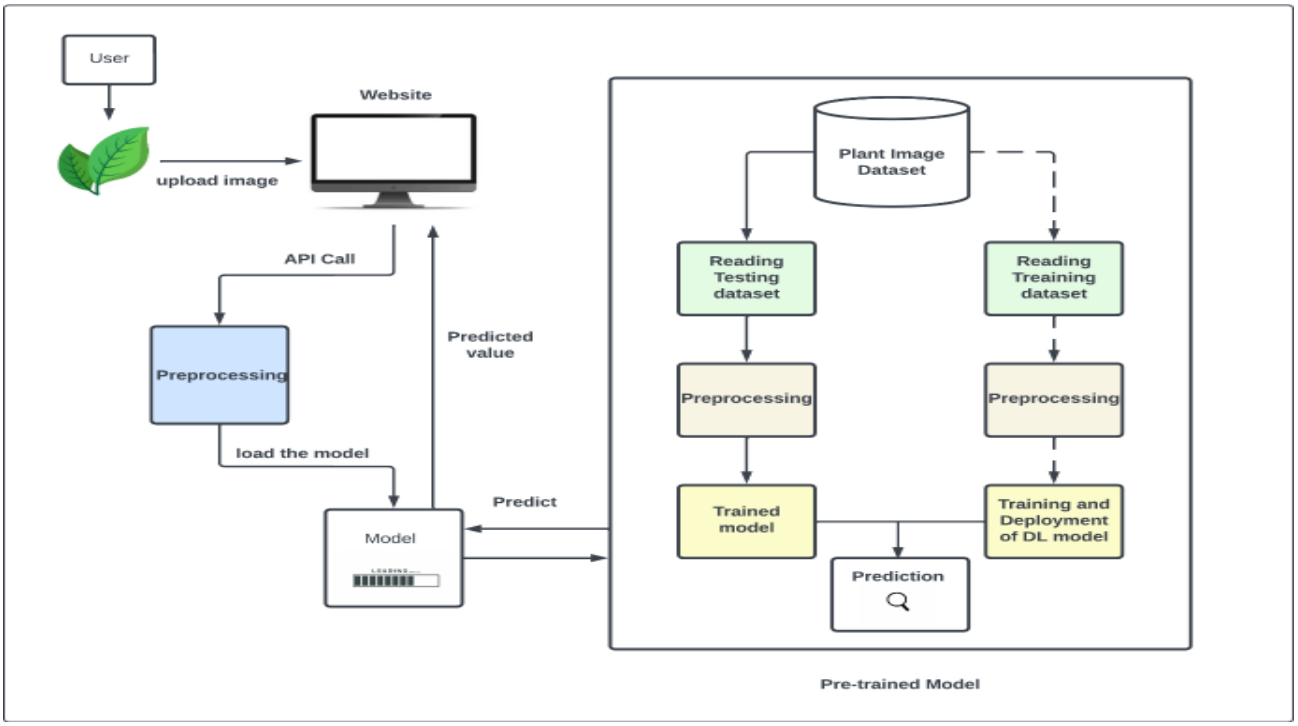

Fig 5. Architecture diagram

to ensure that there is a balanced representation of each plant species; each class contributes a different number of images to this dataset.

Data preprocessing techniques, including resizing, normalization, and augmentation, are applied to standardize and enrich the images, so that the model generalizes to different image conditions. It provides a solid dataset for the training of a robust plant leaf classification model, which will accurately identify plant species based on images of the leaves.

Table I. Dataset description

Dataset	Images	Classes
Medicinal Plants	2438	29

Table II. System Specifications

Operating System	Windows Subsystem for Linux
RAM	16GB DDR4
Processors	AMD Ryzen 4600H Base Frequency: 3 GHz Turbo Clock: Up to 4 GHz
Graphics Card	NVIDIA GeForce GTX 1650; 4GB

The specifications used for the project is mentioned in table II. The specification gives a powerful computing environment for either the training of deep learning models or serving predictions using the Flask server. It allows large datasets, multi core processing capabilities, and GPU acceleration for faster-than-usual computations. This is valuable information, useful in documentation, to understand the hardware environment in which the development and deployment of applications involving machine learning are done.

The software requirements for the developing the model is given in Table III

Table III. Software Requirements for model

Particulars	Version
Python	3.11.5
TensorFlow	2.15.0

B. PREPROCESSING

The dataset is organized, and preprocessing techniques like resizing, normalization, and augmentation are applied. Ensuring balanced augmentation without distorting the original features of the plant leaves posed as a huge challenge. Also, handling imbalanced data distribution across classes required special attention.

Therefore, fine-tuning augmentation parameters and employing techniques like class weighting during model training helped address the imbalanced data issue. Regular monitoring and adjustment of augmentation techniques ensured the preservation of essential leaf features.

C. MODEL

This module involves the development of the deep learning model architecture, including feature extraction and classification layers. Choosing the appropriate pre-trained model architecture and fine-tuning hyperparameters to achieve optimal performance were key challenges. Additionally, selecting the right activation functions and regularization techniques to prevent overfitting was crucial. Extensive experimentation with various pre-trained architectures, such as EfficientNet and ResNet, combined with hyperparameter tuning and regularization techniques, helped identify the most suitable model configuration. Cross-validation and monitoring of training metrics aided in selecting the best-performing model architecture. In the end we found out that using EfficientNetV2 provided us with better results and thus that architecture was opted. There are several layers in the model developed and various paramter. You can refer to that in Table IV and V respectively.

Table IV. Model Layer description

Layer	Output Shape	Number of parameters
KerasLayer (EfficientNet Feature Extractor)	(None, 1280)	20,331,360
Dropout Layer	(None, 1280)	0 (No trainable parameters)
Dense Layer (Output)	(None, 29) (Assuming 29 classes for classification)	37,149

Table V. Parameter description

Total Parameters	20,368,509 (77.70MB)
Trainable Parameters	37,149 (145.11 KB)
Non-trainable Parameters	20,331,360 (77.56MB)

This model utilizes transfer learning with EfficientNet as the feature extractor, followed by a dropout layer to prevent overfitting, and a dense layer for classification. The EfficientNet feature extractor has a large number of non-trainable parameters, while the trainable parameters are mainly in the dense layer for classification. Overall, the model has a total of 20,368,509 parameters, with only

a small portion being trainable, making it efficient for training and inference.

D. MODEL TRAINING

The next step was to train the model on the pre-processed dataset. We leveraged the power of transfer learning, using the weights of the pretrained model and used it for our project. It reduced the training time significantly while maintaining high accuracy. Training deep learning models on large datasets can be computationally intensive and time-consuming, especially without access to high-end hardware resources. Therefore, utilization of hardware acceleration, such as GPUs, and optimizing training parameters, including batch size and learning rate, becomes necessary to help speed up the training process.

The training process involved training the model for 10 epochs (Refer table V). Each epoch represents one complete pass through the entire training dataset. Throughout the training process, both training and validation accuracies were monitored to assess the model's performance and generalization ability.

Table V. Epoch description

Particulars	Value
Number of epochs	10
Duration	22 mins

IV. USER INTERFACE

For the user interface, a web application was developed. We will look into the database, backend and frontend.

A. DATABASE

The details of the plants' names, the binomial nomenclature, the area in which the plant is commonly and predominantly found, the best suitable soil for optimal plant growth and mainly the medicinal benefits of the plant is stored. MongoDB was chosen to store data for this project. It has information about all the 29 classes.

B. BACKEND

Various RESTful APIs were written in the backend for the web application. Flask was used for developing the backend. Being a Python framework, it has a rich environment and support to deal with machine learning and deep learning algorithms which are primarily written in python to be sed in web development. Seamlessly integrating the model with the Flask application, handling concurrent requests efficiently, and optimizing server performance became key challenges. Extensive testing and debugging of the Flask application, optimization of code for efficiency, and deploying the API on scalable infrastructure helped achieve solutions to the problems. Monitoring of server performance and implementing caching mechanisms increased response times and general user experience. The dependencies are given in Table VI.

Table VI. Backend dependencies

Dependencies	Version
flask	2.2.2
flask-cors	4.0.0

This is the interface for handling image upload, preprocessing, model prediction, and sending back prediction values. Users can upload images of plant leaves using the API endpoint. Flask will provide a route to receive image files via HTTP POST requests. The server will save that received image file to a temporary location. The uploaded images undergo preprocessing to get ready for model prediction. It preprocesses the uploaded image to meet the needs of the model's input. Commonly, it involves resizing the image to a standard size (224 x 224), conversion of the image to an array, and normalization of pixel values. The pre-processed image is sent for prediction through the model. In this scenario, the pre-processed image array is fed through the model for making a prediction. The model predicts the class probabilities for each plant species present in the image. The predicted class probabilities are sent to the client as the prediction values. The API returns prediction values to the client in a structured format, usually JSON. The prediction values include the probabilities of each plant species present in the image to interpret.Various small APIs were also written to send data to the frontend.

C. FRONTEND

For the frontend, ReactJS is used. Simple components are made. And drag-zone is used for the drag and drop module to upload the image. On clicking the upload button, the API is called and the fetched data is displayed to the user. Additionally, there are plant details page where the user can view the details of the plants that were predicted.The dependencies required are given in Table VII

Table VII. Frontend dependencies

Dependencies	Version
axios	1.6.7
react	18.2.0
react-dom	18.2.0
react-dropzone	14.2.3
react-router-dom	6.22.3

V. RESULT

The trained model must be evaluated to assesses the trained model's performance on validation and test datasets to ensure its generalization ability. Ensuring that the model performs well on unseen data and identifying potential overfitting or underfitting issues were primary challenges. By using rigorous evaluation metrics, such as

accuracy, precision, comprehensive evaluation of the model could be achieved. Techniques such as early stopping and model checkpointing were employed to prevent overfitting and ensure generalization of the model to new data.

Accuracy metrics provide valuable insights into the model's learning progress and generalization ability. The increasing trend in both training and validation accuracies suggests that the model is effectively learning from the training data and making accurate predictions on unseen data. However, it's essential to evaluate the model's performance on an independent test dataset to obtain a more comprehensive assessment of its accuracy and generalization ability.

A. TRAINING ACCURACY

At the start of training (Epoch 1/10), the training accuracy was approximately 35.29%. This indicates that the model correctly classified around 35.29% of the training data samples. As training progressed, the training accuracy increased steadily with each epoch. By the end of training (Epoch 10/10), the training accuracy reached approximately 96.25%. This indicates a significant improvement in the model's ability to learn from the training data and make accurate predictions. Refer Table VIII

Table VIII. Training accuracy

Epochs	Accuracy
1	35.29%
10	96.25%

B. VALIDATION ACCURACY

The validation accuracy measures how well the model performs on unseen validation data. At the beginning of training (Epoch 1/10), the validation accuracy was approximately 71.31%. Similar to the training accuracy, the validation accuracy showed an increasing trend as training progressed. By the end of training (Epoch 10/10), the validation accuracy reached approximately 95.08%. This indicates that the model not only learned effectively from the training data but also generalized well to unseen validation data.

Table IX. Validation accuracy

Epochs	Accuracy
1	71.31%
10	95.08%

The training and validation accuracy are given in Fig 6

Fig. 6 Training and Validation Accuracy graph

C. COMPARISON WITH OTHER MODELS

Table X. Comparison with other models

Model	**No. of epochs**	**Accuracy**	**Training Time**	**Resource Utilization**
CNN	100	45%	Considerable amount of time	Higher memory usage and computational costs during both training and inference phases.
VGG	30	50%	Required fewer epochs compared to CNN. However, still took a significant amount of time due to its deep architecture.	Demonstrated more efficient resource utilization compared to CNN but still consumed considerable computational resources.
MobileNet	30	70%	Trained faster compared to CNN and VGG due to its lightweight architecture.	Offered improved accuracy while requiring fewer epochs and less memory usage.
EfficientNet with Transfer Learning	10	96%	Significantly reduced training time compared to other architectures	EfficientNet's compound scaling and architecture optimization allowed for achieving high accuracy with fewer parameters and computations,

D. USER INTERFACE

Fig 7. Home page

Fig. 8 Prediction

Fig 9. Prediction Info

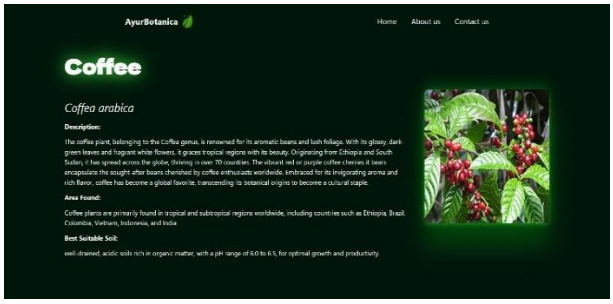

Fig. 10 Plant Info

VI. CONCLUSION

As the new wave of Artificial Intelligence is taking over people's lives making their daily lifestyle easier, incorporating this technology to detect medicinal plants and utilize them would be a more efficient solution to the existing problem of manual detection. Medicinal plants having a plethora of benefits could be easily identified using deep leaning algorithms creating awareness among common people, opening the doors to the potential health benefits of these plants.

We worked with a limited dataset as of the moment. As future enhancement we would like to work with a larger database with a variety of plants to improve the user experience and upscale the utility.

VII. REFERENCES

[1] Yadav, Prakash, T. Pandiaraj, Vikas Yadav, Varsha Yadav, Aina Yadav, and Vyomendra Singh. "Traditional values of medicinal plants, herbs and their curable benefits." *Journal of Pharmacognosy and Phytochemistry* 9, no. 1 (2020): 2104-2106.

[2] Sen, Tuhinadri, and Samir Kumar Samanta. "Medicinal plants, human health and biodiversity: a broad review." *Biotechnological applications of biodiversity* (2015): 59-110.

[3] Mutlag, Wamidh K., Shaker K. Ali, Zahoor M. Aydam, and Bahaa H. Taher. "Feature extraction methods: a review." In *Journal of Physics: Conference Series*, vol. 1591, no. 1, p. 012028. IOP Publishing, 2020.

[4] Sharma, S., and R. Thokchom. "A review on endangered medicinal plants of India and their conservation." (2014): 205-218.

[5] Li, Zewen, Fan Liu, Wenjie Yang, Shouheng Peng, and Jun Zhou. "A survey of convolutional neural networks: analysis, applications, and prospects." *IEEE transactions on neural networks and learning systems* 33, no. 12 (2021): 6999-7019.

[6] Albawi, S., Mohammed, T. A., & Al-Zawi, S. (2017, August). Understanding of a convolutional neural network. In 2017 International Conference on Engineering and Technology (ICET) (pp. 1-6). Ieee.

[7] Kim, Phil, and Phil Kim. "Convolutional neural network." *MATLAB deep learning: with machine learning, neural networks and artificial intelligence* (2017): 121-147.

[8] Putri, Y. A., Djamal, E. C., & Ilyas, R. (2021, March). Identification of Medicinal Plants Leaves Using Convolutional Neural Network. In Journal of Physics:

Conference Series (Vol. 1845, No. 1, p. 012026). IOP Publishing.

[9] Tan, Mingxing, and Quoc Le. "Efficientnetv2: Smaller models and faster training." In *International conference on machine learning*, pp. 10096-10106. PMLR, 2021.

[10] Weiss, Karl, Taghi M. Khoshgoftaar, and DingDing Wang. "A survey of transfer learning." *Journal of Big data* 3 (2016): 1-40.

[11] Gholamalinezhad, Hossein, and Hossein Khosravi. "Pooling methods in deep neural networks, a review." *arXiv preprint arXiv:2009.07485* (2020).

[12] Ma, Wei, and Jun Lu. "An equivalence of fully connected layer and convolutional layer." *arXiv preprint arXiv:1712.01252* (2017).

[13] Abdollahi, Jafar. "Identification of medicinal plants in ardabil using deep learning: identification of medicinal plants using deep learning." In *2022 27th International Computer Conference, Computer Society of Iran (CSICC)*, pp. 1-6. IEEE, 2022.

[14] Dileep, M. R., and P. N. Pournami. "AyurLeaf: a deep learning approach for classification of medicinal plants." In *TENCON 2019-2019 IEEE Region 10 Conference (TENCON)*, pp. 321-325. IEEE, 2019..

[15] Malik, Owais A., Nazrul Ismail, Burhan R. Hussein, and Umar Yahya. "Automated real-time identification of medicinal plants species in natural environment using deep learning models—a case study from Borneo Region." *Plants* 11, no. 15 (2022): 1952.