AYURBOTANICA – IDENTIFICATION OF MEDICINAL PLANTS

A PROJECT REPORT

Submitted by

A. SATYANARAYANA   (2116210701235)
K. S. J. SNEHA   (2116210701253)

in partial fulfillment for the award of

the degreeof

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

RAJALAKSHMI ENGINEERING

COLLEGEANNA UNIVERSITY,

CHENNAI

MAY 2024

RAJALAKSHMI ENGINEERING COLLEGE, CHENNAI

BONAFIDE CERTIFICATE

Certified that this Thesis titled **"AYURBOTANICA – IDENTIFICATION OF MEDICINAL PLANTS"** is the bonafide work of "**A. SATYANARAYANA (2116210701235), K.S.J.SNEHA (2116210701253) and** who carried out the work under my supervision. Certified further that to the best of my knowledge the work reported herein does not form part of any other thesis or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE

Dr. Rakesh Kumar M, M.E., Ph.D.,

Project Coordinator

Assistant Professor (SG)

Department of Computer Science and
Engineering

Rajalakshmi Engineering College

Chennai - 602 105

Submitted to Project Viva-Voce Examination held on_____

Internal Examiner   **External Examine**r

ACKNOWLEDGMENT

First, we thank the almighty god for the successful completion of the project. Our sincere thanks to our chairman **Mr. S. Meganathan B.E., F.I.E.,** for his sincere endeavor in educating us in his premier institution. We would like to express our deepgratitude to our beloved Chairperson **Dr. Thangam Meganathan Ph.D.,** for her enthusiastic motivation which inspired us a lot in completing this project and Vice Chairman **Mr. Abhay Shankar Meganathan B.E., M.S.,** for providing us with the requisite infrastructure.

We also express our sincere gratitude to our college Principal,

Dr. S. N. Murugesan M.E., PhD., and **Dr. P. KUMAR M.E., PhD., Director computing and information science, and Head Of Department of Computer Science and Engineering** and our project coordinator **Dr. Rakesh Kumar M, M.E., Ph.D.,** for his encouragement and guiding us throughout the project towards successful completion of this project and to our parents, friends, all faculty members and supporting staffs for their direct and indirect involvement in successful completion of the project for their encouragement and support.

A.SATYANARAYNA

K.S.J.SNEHA

iii

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. Moreover, preserving these plants for the future generations is also crucial. Since people are less aware of these medicinal plants, they tend to be negligent towards these plants. If people can be made aware of the plants, they are surrounded with along with their health benefits it could push them to start adopting these ayurvedic herbs instead of drugs with side effects. Going back to our roots can actually help solve many problems. 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. Moreover, lack of trained experts In medicinal plants are very few in number. In order to promote medicinal plants, AyurBotanica can help play a role by actively engaging its users to explore the plants around them.

LIST OF TABLES

LIST OF FIGURES

TABLE OF CONTENTS

CHAPTER 1

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, utilizing 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, detection 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

1.1 PROBLEM STATEMENT

The AyurBotanica project aims to address the challenge of accurately identifying various medicinal plants from images. This involves developing a deep learning model using EfficientNetV2 with transfer learning to ensure high accuracy in plant identification. The project requires the creation of a user-friendly web interface using ReactJS and Flask, allowing users to upload images of plants for identification. The backend will manage data using MongoDB. The goal is to provide users with detailed information about identified plants, including botanical names and medicinal properties thereby supporting botanical research, healthcare, and conservation efforts by making plant identification accessible and efficient.

1.2 SCOPE OF THE WORK

AyurBotanica which utilizes EfficientNetV2 architecture leveraging the power of transfer learning offers the users a chance to learn more about the medicinal plants around them and its potential health benefits. Anyone can use this application to get to know about the medicinal plants around them. It helps us by giving us an opportunity to prevent losing these to extinction by making people aware of the plants around them. It can be used by Traditional Medicinal Practitioners to verify the plants. Pharmaceutical companies whose medicines majorly depends on herbs and plants can use it for easy and quick identification of these medicinal plants. algorithm fine-tuning, the simplicity and adaptability of the game present exciting opportunities for future enhancements.

1.3 AIM AND OBJECTIVES OF THE PROJECT

The primary goal of this project is to develop a deep learning model to identify and classify the medicinal plants. This can reduce the lack of knowledge that people have when it comes to herbs and plants. This is done so that everyone can benefit from the amazing health benefits of these medicinal plants.

It reduces the time taken by the manual laborious work done by traditional method of manual identification of the plants by experts. Also, another aim of this project would be to we need to create a user-friendly user interface that the users can use the learn more about the medicinal plants around them.

CHAPTER 2
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]

2.1. 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.
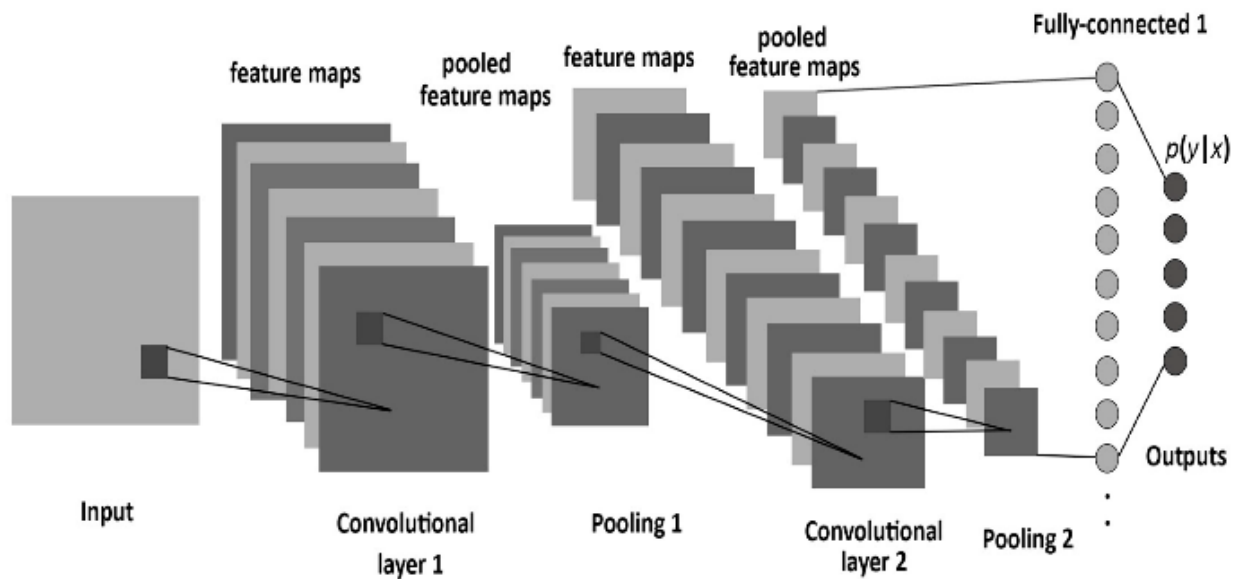
The various layers in CNN are depicted in Fig. 2.

Fig. 2. Layers in CNN

The general outline of the layers is as follows:

2.1.1. 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

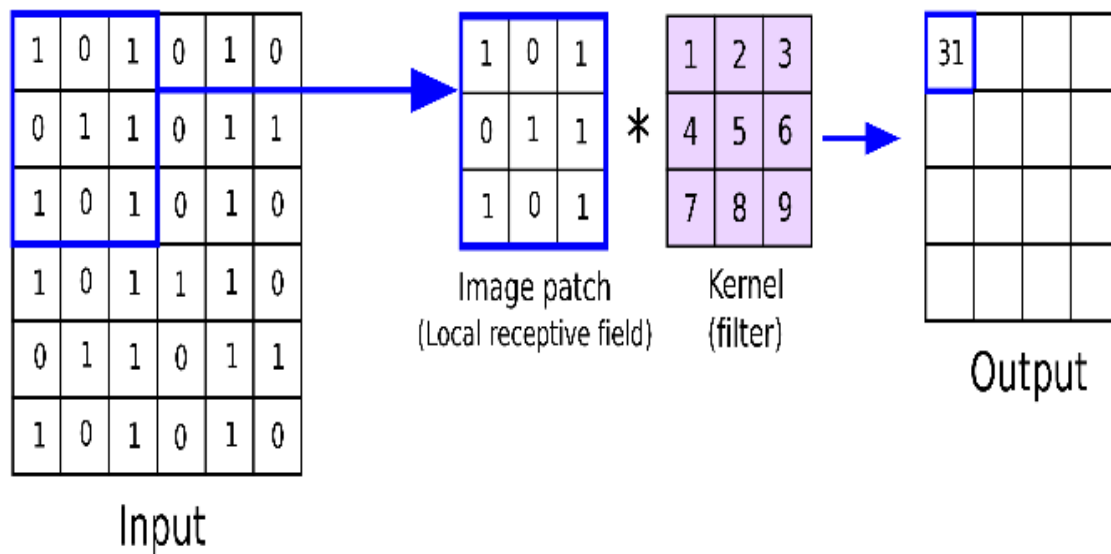## 2.1.2. POOLING LAYER

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
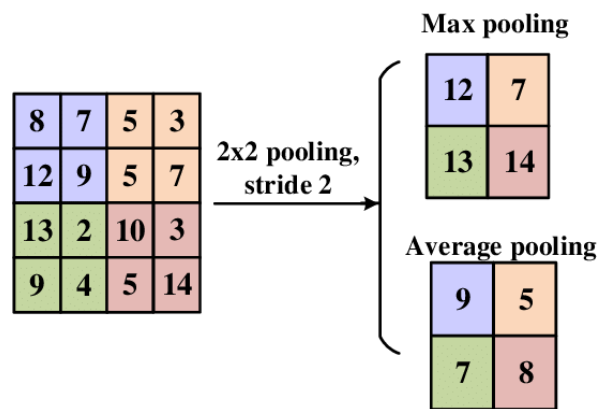
Fig. 4. Pooling Layer

2.1.3. FULLY CONNECTED LAYER:

Fully connected layers (dense layers) are essential in neural networks, especially for tasks like classification and regression. Each neuron in a fully connected layer connects to every neuron in the preceding layer, allowing it to learn global patterns.[12]

Structure and Functionality

Fully connected layers take the flattened input from previous layers and process it using weights and biases:

$y=f(Wx+b)$

where y is the output,

W is the weight matrix,

x is the input,

b is the bias, and

f is the activation function.

The SoftMax activation function is often used in the final layer for classification, converting raw scores into probabilities.

Role in Neural Networks

Typically appearing at the end of convolutional neural networks (CNNs), fully connected layers combine features extracted by convolutional and pooling layers to make final predictions. [21-23]

2.2. 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.

Mask Defect Detection:

An improved EfficientNetV2 model combined with attention mechanisms and advanced activation functions achieves high accuracy in detecting complex mask defects, showcasing its strong feature extraction capabilities [19]

Remote Sensing Image Classification:

The CA-EfficientNetV2 model, which incorporates coordinate attention, enhances the classification accuracy of remote sensing images, achieving superior performance on datasets like UC Merced. [20]

2.3. 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]

2.4. FEATURE EXTRACTION

Feature extraction is a fundamental process in various fields such as image processing, pattern recognition, and machine learning. It involves reducing the dimensionality of data while preserving important information for classification, diagnosis, or recognition tasks. This survey explores different feature extraction techniques, their applications, and comparative performance based on recent literature.

Biomedical Image Processing:

Feature extraction methods include structural and graph descriptors, transformed and non-transformed signal characteristics, such as principal component analysis (PCA), discrete Fourier transform (DFT), and discrete cosine transform (DCT). These methods are crucial for reducing the dimensionality of pattern representations, thereby enhancing classification accuracy and reducing computational complexity [18]

General Feature Extraction Techniques:

Techniques like Sobel and Canny edge detection, Hough transformation, and texture analysis are commonly used for extracting features from images. These methods help in identifying edges, lines, and textures which are critical for image processing applications [16]

Comparative Analysis of Feature Extraction Methods:

A study comparing geometric, statistical, texture, and color features found that the choice of feature extraction method significantly affects the performance depending on the application. For example, face and plant image classification benefit from different feature sets [17]

.

CHAPTER 3
SYSTEM DESIGN

3.1 GENERAL

In this section, we would like to show how the general outline of how all the components end up working when organized and arranged together. It is further represented in the form of a flow chart below.
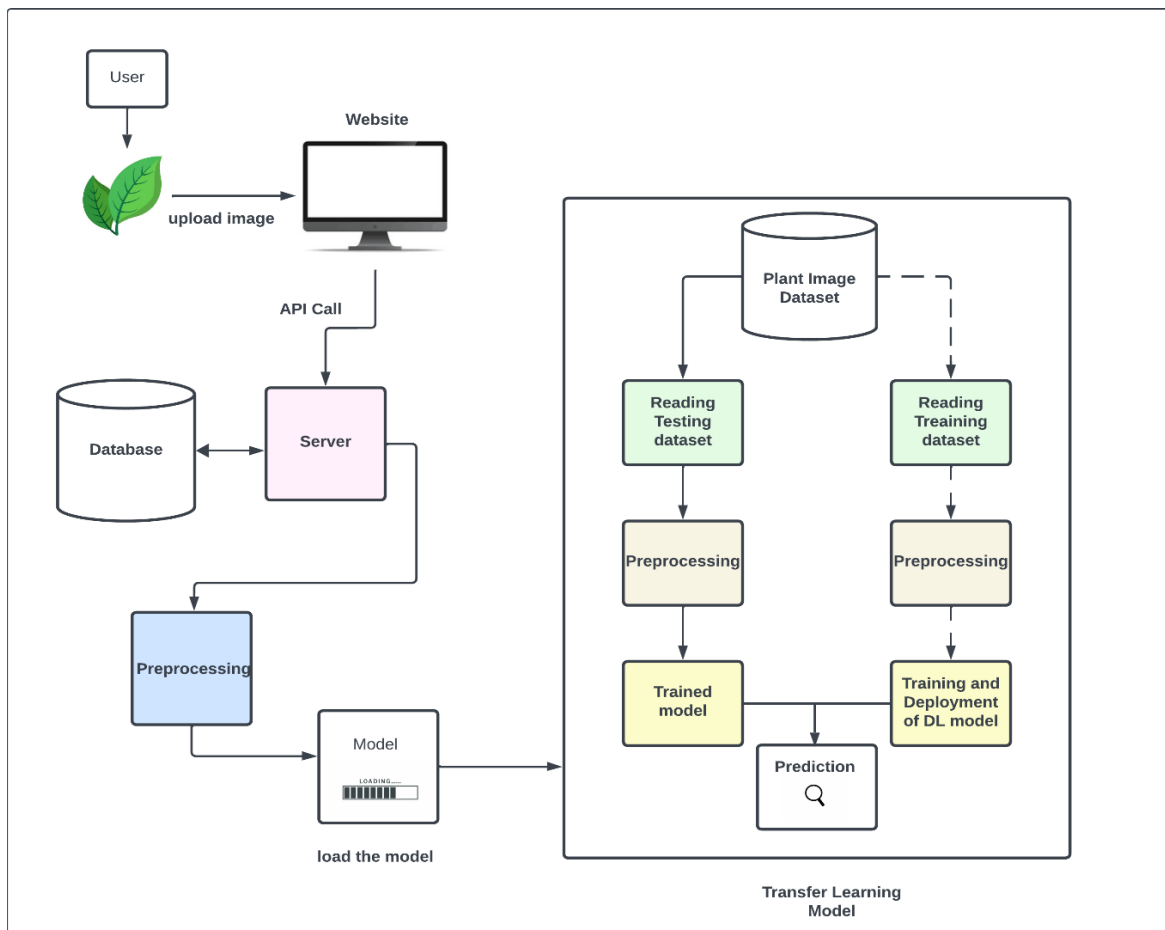
3.2 SYSTEM ARCHITECTURE DIAGRAM

Fig. 5. System Architecture

3.2.1. USER INTERFACE:

The frontend of AyurBotanica serves as the primary user interface, designed with ReactJS to ensure a responsive and interactive user experience. Users initiate the plant identification process by uploading an image of the plant through the application's interface. Upon image upload, the frontend sends an HTTP POST request to the backend API endpoint using Axios or Fetch API.

Image Upload: Users can drag and drop or select an image file from their device.

Result Display: Once the backend returns the prediction results, the frontend dynamically displays the top 5 predictions with confidence scores and detailed plant information.

3.2.2 PREPROCESSING:

The preprocessing step is crucial for ensuring the uploaded images are compatible with the EfficientNetV2 model. The preprocessing pipeline includes:

Image Resizing: Images are resized to the required dimensions (e.g., 224x224 pixels) to match the model's input size.

Normalization: Pixel values are normalized to a specific range (e.g., 0 to 1) to align with the model's training data.

Data Augmentation: Techniques such as rotation, flipping, and scaling might be applied to enhance model robustness by simulating various plant orientations and conditions.

Format Conversion: Images are converted to the required format (e.g., RGB) if necessary.

3.2.3 EFFICIENTNETV2 MODEL WITH TRANSFER LEARNING:

EfficientNetV2 is a state-of-the-art convolutional neural network designed for high performance and efficiency. In AyurBotanica, this model has been fine-tuned using transfer learning:

Pre-trained Weights: Leveraging pre-trained weights on a large dataset (e.g., ImageNet) to utilize the learned features of EfficientNetV2.

Custom Dataset: The model is further trained on a custom dataset specific to medicinal plants, enabling it to identify a wide variety of plant species.

Training Process: The model training involves fine-tuning layers to adapt the pre-trained model to the specific task of plant identification, optimizing for accuracy and speed.

Prediction: After preprocessing, the image is fed into the model, which outputs the top 5 predictions along with confidence scores, indicating the likelihood of each prediction being correct.

3.2.4 DATABASE:

The MongoDB database serves as a comprehensive repository for medicinal plant information. The database schema includes:

Binomial Nomenclature: Scientific names of plants.

Geographic Distribution: Regions where the plants are commonly found.

Optimal Soil Conditions: Preferred soil types and conditions for plant growth.

Medicinal Properties: Detailed information on the medicinal uses and active compounds in the plants.

The Flask backend API handles CRUD operations to manage this data and provides endpoints for retrieving plant information based on the model's predictions.

3.2.5 PREDICTION:

The prediction process integrates model results with database information to deliver a detailed plant profile to the user:

Top 5 Predictions: The model's top 5 predictions are sent back to the frontend, each with an associated confidence score.

Detailed Plant Profiles: For each predicted plant, the frontend fetches detailed information from the MongoDB database via the Flask API.

User Interface: The frontend displays the predictions and detailed plant profiles, allowing users to view and compare multiple potential matches.

3.3. SYSTEM SPECIFICATIONS

3.3.1 HARDWARE SPECIFICATIONS

The hardware requirements for the system are as follows

Table 1. 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

3.3.2 SOFTWARE REQUIREMENTS

The software requirements and dependencies installed for both the model and the web application is mentioned below

Table 2. Software Requirements

Dependencies	Version
Python	3.11.5
TensorFlow	2.15.0
flask	2.2.2
flask-cors	4.0.0
axios	1.6.7
react	18.2.0
react-dom	18.2.0
react-dropzone	14.2.3
react-router-dom	6.22.3

CHAPTER 4
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, 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. 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.

4.1. 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 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 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. Refer Table 3.

Table 3. Dataset description

Dataset	Images	Classes
Medicinal Plants	2438	29
Training	1706	29
Testing	366	29
Validation	366	29

CHAPTER 5
MODULE DESCRIPTION

5.1. 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.

5.2. 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 parameter.

You can refer to that in Table 4 and 5 respectively.

Table 4. 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 5. 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.

5.3. 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 6. Epoch description

Particulars	Value
Number of epochs	10
Duration	22mins

5.4. USER INTERFACE

For the user interface, a web application was developed to streamline the identification process of medicinal plants. This application integrates a robust backend, developed using Flask, with MongoDB for efficient data storage and management. The frontend, built with ReactJS, provides an intuitive platform for users to upload images of plants. The deep learning model, leveraging EfficientNetV2 with transfer learning, processes these images to identify the plants and display detailed information about their botanical characteristics and medicinal properties

5.4.1 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.

5.4.2. 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.

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.

5.4.3. 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.

CHAPTER 6
RESULTS AND OUTPUT

6.1. RESULTS

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.

6.1.1. 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 7

Table 7. Training accuracy

Epochs	Accuracy
1	35.29%
10	96.25%

6.1.2. 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. Refer Table 8

Table 8. Validation accuracy

Epochs	Accuracy
1	71.31%
10	95.08%

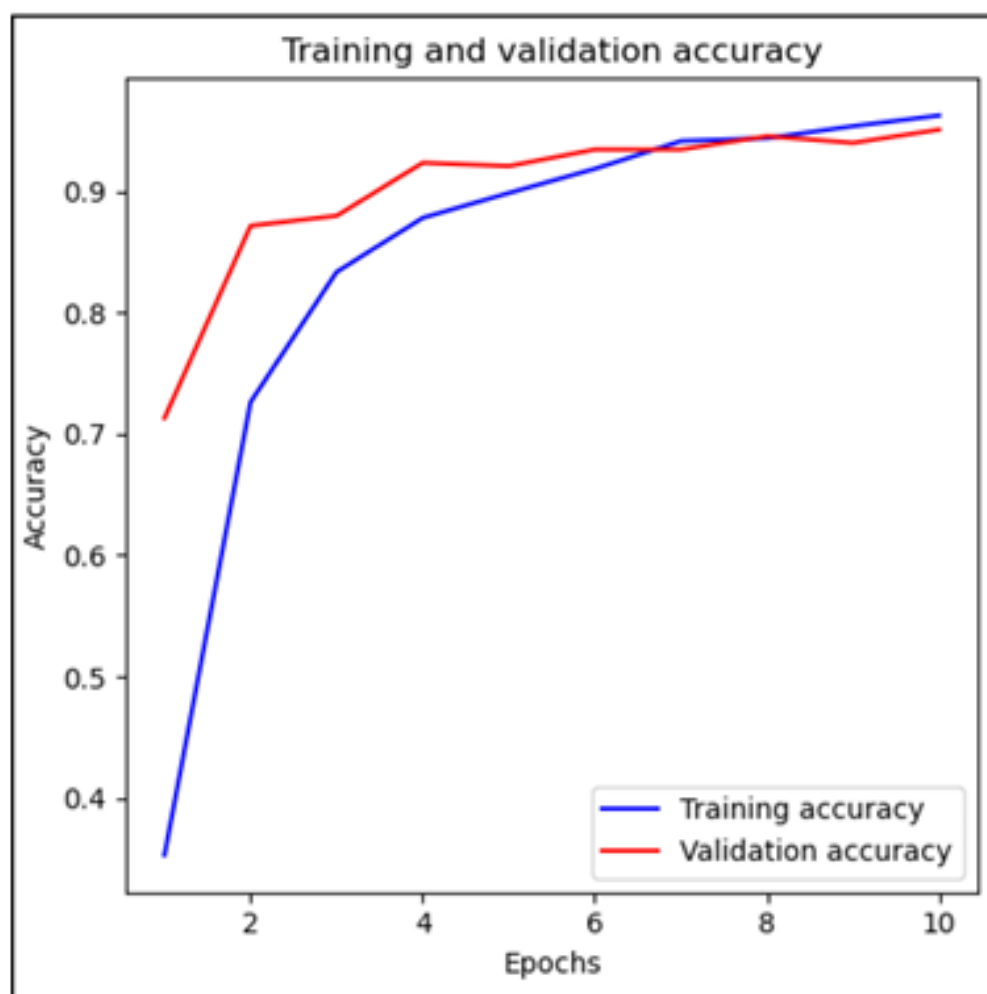The training and validation accuracy are given in Fig 6

Fig. 6 Training and Validation Accuracy graph

6.2. COMPARISON WITH OTHER MODELS

Table 9. 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 due to its efficient architecture and transfer learning.	EfficientNet's compound scaling and architecture optimization allowed for achieving high accuracy with fewer parameters and computations,

6.3. OUTPUT

Fig 7. Home page

The Home page contains the title of our project and a small description of what it does. It is where the user can upload the image of the plant to get the prediction of what the plant could potentially be. There is a drag and drop module. The user can either upload manually or drag and drop the image.

Fig. 8. Prediction page

After the user uploads the image and clicks on predict, a pop up is displayed where the the top 5 prediction values are displayed. The user can either click "Ok" and exit this pop up or click "Know more" to view the plant details.

Fig 9. Prediction Info

When the user clicks on "Know more", It is directed to this page. The top three plant cards are displayed to the user. The last two values are discarded due to the insignificant prediction values. When the user hovers over the cards, the plant name and prediction value is visible. If the user clicks on any of the card, its details will be shown

Fig. 10 Plant Info

The plant information is fetched from the database and displayed to the user. All the necessary information like binomial nomenclature, description, Area the plsnt is usually found in, the best suitable soil for optimal plant growth, and its medicinal values are displayed to the user.

CHAPTER 6
CONCLUSION AND FUTURE
ENHANCEMENT

6.1 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. It is good to embrace change, especially in the current world of ever-growing technology, but we shouldn't forget our roots. It is important to be aware of its importance and to which extent it can help us. By using a modern technology like deep learning, we can make use of our traditional medicines while also catching with the trend. With Artificial Intelligence being the buzz word , harnessing its power to identify medicinal plants could be an amazing and intuitive solution to our current problem.

6.2 FUTURE ENHANCEMENT

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. We can train it with various local area plant dataset to give us results which are specific to a particular area only. We can incorporate map functionality to locate areas where the plants were discovered.

APPENDIX

SOURCE CODE:

BACKEND:

```python
from flask import Flask, request, jsonify
from flask_cors import CORS
from werkzeug.utils import secure_filename
import os
from pymongo import MongoClient
from dotenv import load_dotenv
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
import tensorflow_hub as hub
import numpy as np
import os


app = Flask(_name_)
CORS(app)
load_dotenv()
MONGO_URI = os.getenv('MONGO_URI')
DB_NAME = os.getenv('DB_NAME')
COLLECTION_NAME = os.getenv('COLLECTION_NAME')


try:
    client = MongoClient(MONGO_URI)
    db = client[DB_NAME]
    collection = db[COLLECTION_NAME]
```

```python
    print("Connected to MongoDB successfully!")

except Exception as e:
    print("Error connecting to MongoDB", e)

UPLOAD_FOLDER = 'uploads'
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER
model=load_model("../Model/AyurBotClassEf.h5",custom_objects={'KerasLayer':hub.KerasLayer})
classes=os.listdir("/mnt/d/Academics/Projects/DL/AyurBotanica-TransferLearningModel/dataset/")
n=len(classes)
print(n)
label_ind={}

for i in range(n):
    label_ind[i]=classes[i]

def preprocess_image(img_path):
    img=image.load_img(img_path,target_size=(224,224))
    img_array=image.img_to_array(img)
    img_array=np.expand_dims(img_array,axis=0)
    img_array = img_array / 255.0
    return img_array

@app.route('/predict', methods=['POST'])
def predict():
```

```python
    if "image" not in request.files:
        return jsonify({"error":"No images uploaded"})


    img_file=request.files["image"]
    img_path="temp_path.jpeg"
    img_file.save(img_path)
    img_array=preprocess_image(img_path)


try:

    predictions=model.predict(img_array)
    print(predictions)
    os.remove(img_path)
    predList=predictions.tolist()
    # print(predList)
    l=[]
    for i in range(n):
        temp=[predList[0][i],label_ind[i]]
        l.append(temp)
    l.sort()
    l.reverse()
    ans=[]
    for i in range(5):
        ans.append(l[i])

    # os.remove(img_path)
    return jsonify({"predictions":ans})
```

```python
    except Exception as e:
        os.remove(img_path)
        return jsonify({"error":str(e)})



@app.route("/fetch", methods=['POST'])
def fetch():

    try:
        plant_name = request.json.get('plant_name')
        # print(plant_name)


        plant_detail = collection.find_one({'name': plant_name})
        # print(plant_detail)


        if plant_detail:
            plant_detail['_id'] = str(plant_detail['_id'])  # Convert ObjectId to string
            return jsonify({'plant_detail': plant_detail})
        else:
            return jsonify({'error': 'Plant not found'}), 404


    except Exception as e:
        print("Error fetching details", e)
        return jsonify({"error": str(e)}), 400


if _name_ == '_main_':
    app.run(debug=False)
```

IMPORTS MODEL:

```python
import os
import numpy as np
import pandas as pd
import tensorflow as tf
from keras import layers
import tensorflow_hub as hub
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.callbacks import EarlyStopping, ReduceLROnPlateau
from tensorflow.keras.layers import Dense, GlobalAveragePooling2D,Dropout
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.models import Model
from sklearn.model_selection import train_test_split
from tensorflow.keras.preprocessing.image import img_to_array, load_img
from keras import Sequential
```

DATASET LABELLING:

```python
dataset="dataset"
img_height,img_width=224,224
batch_size=32


classes=os.listdir(dataset)
print(classes)


n=len(classes)
# all the images and their corresponding labels
```

```
plant_images=[]
labels=[]

for plant in classes:
    images=os.path.join(dataset,plant)
    for image in os.listdir(images):
        plant_images.append(os.path.join(images,image))
        labels.append(plant)
print(plant_images)
print(labels)
```

TRAIN TEST SPLIT:

```
x_train,x_test,y_train,y_test=train_test_split(plant_images,labels,test_size=0.3,random_st
ate=42)

x_test, x_val, y_test, y_val = train_test_split(x_test, y_test, test_size=0.5,
random_state=42)

# Create DataFrames for training, validation, and testing
train_data = pd.DataFrame({"plant_images": x_train, "labels": y_train})
val_data = pd.DataFrame({"plant_images": x_val, "labels": y_val})
test_data = pd.DataFrame({"plant_images": x_test, "labels": y_test})
```

DATA AUGMENTATION:

```
train_aug=ImageDataGenerator(
    rescale=1./255,
```

```
    rotation_range=20,

    shear_range=0.2,

    zoom_range=0.2,

    horizontal_flip=True,

    vertical_flip=True,

    fill_mode="nearest"

)

test_aug=ImageDataGenerator(

    rescale=1./255

)

val_aug=ImageDataGenerator(

    rescale=1./255

)


train_gen=train_aug.flow_from_dataframe(

    train_data,

    x_col="plant_images",

    y_col="labels",

    target_size=(img_height,img_width),

    batch_size=batch_size,

    class_mode="categorical",

    shuffle=True

)


test_gen=test_aug.flow_from_dataframe(

    test_data,

    x_col="plant_images",
```

```python
    y_col="labels",

    target_size=(img_height,img_width),

    batch_size=batch_size,

    class_mode="categorical"

)


val_gen=val_aug.flow_from_dataframe(

    val_data,

    x_col="plant_images",

    y_col="labels",

    target_size=(img_height,img_width),

    batch_size=batch_size,

    class_mode="categorical"

)
```

MODEL:

```python
feature_ext_model=

"https://tfhub.dev/google/imagenet/efficientnet_v2_imagenet21k_ft1k_s/feature_vector/2
"


pretrained_mod =hub.KerasLayer(

    feature_ext_model, input_shape=(img_height, img_width, 3), trainable=False

)


model=tf.keras.Sequential([

    pretrained_mod,

    Dropout(0.3),
```

```
    Dense(n,activation="softmax")

])


model.compile(optimizer="adam",loss="categorical_crossentropy",metrics=["accuracy"])
model.summary()
```

TRAINING:

```
history=model.fit(

    train_gen,

    epochs=10,

    validation_data=val_gen,

)

model.save("AyurBotClassEf.h5")
```

TESTING:

```
# modelTry = tf.keras.models.load_model("AyurBotClassEf.h5")

modelTry=model

test_loss, test_accuracy = modelTry.evaluate(test_gen)


print("Test Loss:", test_loss)

print("Test Accuracy:", test_accuracy)
```

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.

[16] Aichert, André. "Feature extraction techniques." In *Camp medical seminar ws0708*, pp. 1-8. 2008.

[17] 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.

[18] Meyer-Bäse, Anke. *Pattern Recognition and Signal Analysis in Medical Imaging*. Academic Press, 2004.

[19] Lamei, Liu, Fang Junjie, Huang Huiling, Zhang Yongjian, and Han Jun. "Mask Defect Detection Algorithm Based on Improved EfficientNetV2." In *2022 International Symposium on Advances in Informatics, Electronics and Education (ISAIEE)*, pp. 507-510. IEEE, 2022.

[20] Wang, Zengkun, Yang Cao, Hongfei Yu, Caihua Sun, Xuejian Chen, Zhanggen Jin, and Weili Kong. "Scene classification of remote sensing images using EfficientNetV2

with coordinate attention." In *Journal of Physics: Conference Series*, vol. 2289, no. 1, p. 012026. IOP Publishing, 2022.

[21] Tan, M., & Le, Q. V. (2021). EfficientNetV2: Smaller Models and Faster Training. EfficientNetV2: Smaller Models and Faster Training.

[22] Jeon, H.-J., & Jeon, J. (2022). Efficient Training of EfficientNetV2-S Using AdaBelief Optimizer. Efficient Training of EfficientNetV2-S Using AdaBelief Optimizer.

[23] Lamei, L., Junjie, F., Huiling, H., Yongjian, Z., & Jun, H. (2022). Mask Defect Detection Algorithm Based on Improved EfficientNetV2. Mask Defect Detection Algorithm Based on Improved EfficientNetV2.