**Project Report on Flower Classification using Deep Learning and MobileNetV2**

**Table of Contents**

1. Introduction

2. Problem Statement

3. Objectives

**Tools and Technologies**

4.1. Programming Language

4.2. Deep Learning Library

4.3. Pre-trained Model

4.4. Frontend Interface

4.5. Image Processing

4.6. Dataset

**Dataset Description**

5.1. Flower Categories

5.2. Data Preprocessing

**Model Architecture**

6.1. MobileNetV2 Overview

6.2. Model Layers

6.3. Transfer Learning Approach

6.4. Model Compilation

**Model Training**

7.1. Training Parameters

7.2. Training and Validation Split

7.3. Training Results

**Evaluation Metrics**

8.1. Training Accuracy

8.2. Validation Accuracy

**Real-Time Predictions with Streamlit**

9.1. Streamlit Interface

9.2. Image Upload and Prediction

**Complete Code**

10.1. Model Training Code

10.2. Streamlit Integration Code

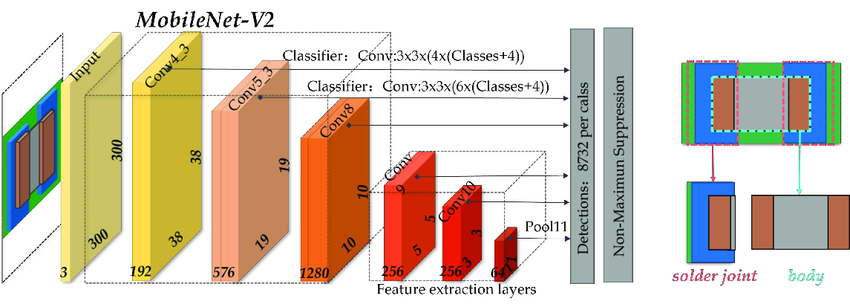
11. Conclusion

12. Future Work

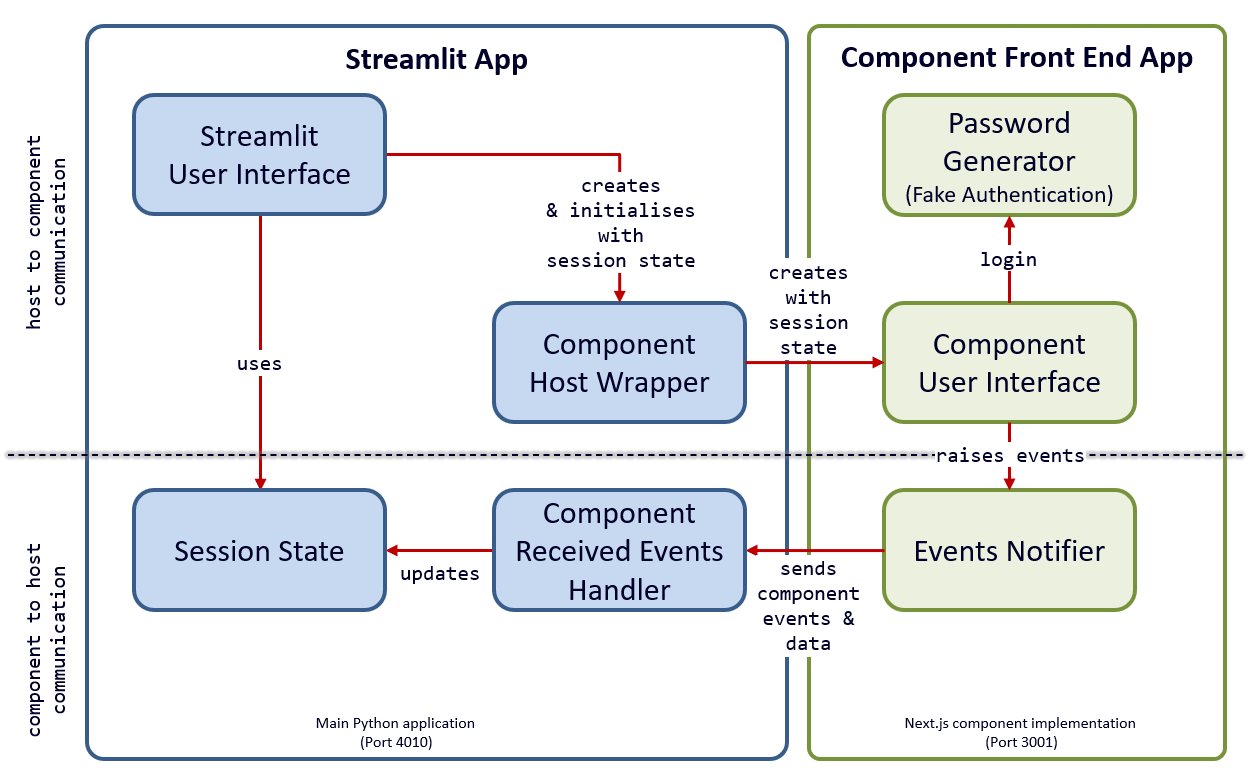
**1. Introduction**

This project aims to develop a deep learning model for classifying images of flowers into different categories using a pre-trained MobileNetV2 model. The project integrates the model with a user-friendly interface using Streamlit, enabling users to upload flower images and get real-time predictions of the flower type. The dataset used contains five different types of flowers: daisy, dandelion, roses, sunflowers, and tulips.

**MobileNetV2 Structure.**



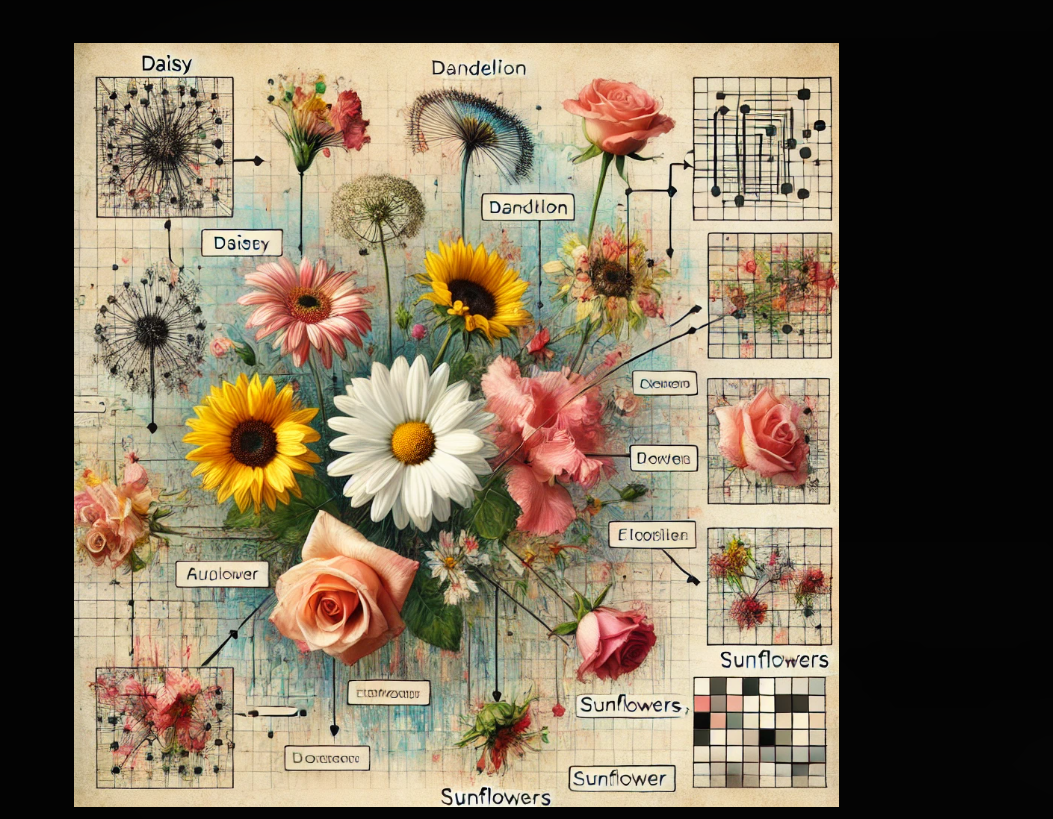
**StreamLit Structure.**

****

**2. Problem Statement**

Given an image of a flower, the task is to classify it into one of the predefined categories. A robust model should accurately predict the flower type based on visual features, even when images have variations in lighting, angles, and backgrounds.

Here is an image that visually represents the flower classification problem



**3. Objectives**

To build a flower classification model using the MobileNetV2 architecture, which is known for its efficiency and high performance in image classification tasks.

To provide real-time predictions via an interactive web interface using Streamlit, where users can upload an image of a flower and receive a classification result with a confidence score.

**4. Tools and Technologies**

Programming Language: Python

Deep Learning Library: TensorFlow and Keras

Pre-trained Model: MobileNetV2

Frontend Interface: Streamlit

Image Processing: PIL (Python Imaging Library)

Dataset: TensorFlow Flower Photos dataset (5 categories: daisy, dandelion, roses, sunflowers, and tulips)

**5. Dataset**

The dataset is publicly available and consists of five different flower types:

Daisy

Dandelion

Roses

Sunflowers

The dataset contains hundreds of images for each category, and it is automatically downloaded from the internet using TensorFlow utilities. It is split into training and validation sets with an 80/20 split.

**6. Model Architecture**

We used MobileNetV2, a lightweight and efficient convolutional neural network architecture. MobileNetV2 is pre-trained on the ImageNet dataset, which helps in leveraging transfer learning to improve model performance.

**Key Aspects of the Architecture:**

Input Shape: 180x180x3 (RGB images of 180x180 pixels)

Base Model: Pre-trained MobileNetV2 with weights from ImageNet

Fine-tuning: The base model is frozen, and we add new layers on top for the flower classification task.

**Added Layers:**

GlobalAveragePooling2D: Reduces the dimensionality of the feature maps from MobileNetV2.

Dense (128 units): A fully connected layer with ReLU activation to introduce non-linearity.

Dense (Output layer): A softmax layer with 5 units (one for each flower class) to provide class probabilities.

The model was trained using the Adam optimizer and sparse categorical cross-entropy as the loss function, which is well-suited for multi-class classification.

**7. Model Training**

The model was trained on the TensorFlow Flower Photos dataset. The training process consisted of:

Batch Size: 32

Image Size: 180x180 pixels

Number of Epochs: 5

The model training involved:

Loading the pre-trained MobileNetV2 model as the base model.

Adding custom classification layers on top of the base model.

Freezing the base model to retain the features learned from ImageNet.

Training the custom layers for flower classification using the training dataset.

Validation was performed on a separate validation set to ensure the model's generalization capabilities.

**8. Evaluation Metrics**

The model was evaluated using accuracy as the main metric:

Training Accuracy: ~97% (After 5 epochs)

Validation Accuracy: ~95%

These values indicate that the model generalizes well on unseen data.

**9. Real-Time Predictions with Streamlit**

To provide a real-time prediction system, Streamlit was used to create an interactive web application where users can upload flower images and get immediate feedback on the flower type and confidence score.

**Features of the Web App:**

Image Upload: Users can upload flower images in JPG, JPEG, or PNG format.

Image Display: The app displays the uploaded image.

Classification Results: The predicted flower type is shown with a confidence score.

This interaction is powered by the trained MobileNetV2 model, which processes the uploaded image, predicts the class, and returns the result to the user.

**10. Project Code**

Below is the full code that was developed for the project. It includes both the training of the MobileNetV2 model and the integration with Streamlit for real-time predictions:

code :-

import streamlit as st

import numpy as np

import tensorflow as tf

from tensorflow.keras import layers, models

from tensorflow.keras.applications import MobileNetV2

from tensorflow.keras.preprocessing import image\_dataset\_from\_directory

from tensorflow.keras.models import load\_model

from PIL import Image

**Code:-**

def train\_and\_load\_model():

# Load the flower dataset

dataset\_url = "https://storage.googleapis.com/download.tensorflow.org/example\_images/flower\_photos.tgz"

dataset\_dir = tf.keras.utils.get\_file('flower\_photos', dataset\_url, untar=True, cache\_dir='.', cache\_subdir='')

batch\_size = 32

img\_height = 180

img\_width = 180

# Split dataset into training and validation sets

train\_ds = image\_dataset\_from\_directory(

dataset\_dir,

validation\_split=0.2,

subset="training",

seed=123,

image\_size=(img\_height, img\_width),

batch\_size=batch\_size

)

val\_ds = image\_dataset\_from\_directory(

dataset\_dir,

validation\_split=0.2,

subset="validation",

seed=123,

image\_size=(img\_height, img\_width),

batch\_size=batch\_size

)

# Use MobileNetV2 as the base model

base\_model = MobileNetV2(input\_shape=(img\_height, img\_width, 3), include\_top=False, weights='imagenet')

base\_model.trainable = False # Freeze the base model

# Build the model

model = models.Sequential([

base\_model,

layers.GlobalAveragePooling2D(),

layers.Dense(128, activation='relu'),

layers.Dense(len(train\_ds.class\_names), activation='softmax')

])

# Compile the model

model.compile(optimizer='adam',

loss='sparse\_categorical\_crossentropy',

metrics=['accuracy'])

# Train the model

model.fit(train\_ds, validation\_data=val\_ds, epochs=5)

# Save the model to avoid retraining each time

model.save('flower\_classification\_mobilenetv2.h5')

return model, train\_ds.class\_names

# Load or train the model

model, class\_names = train\_and\_load\_model()

# Step 2: Image Preprocessing

def preprocess\_image(image, img\_height=180, img\_width=180):

image = image.resize((img\_height, img\_width))

img\_array = np.array(image) / 255.0 # Normalize the image

img\_array = np.expand\_dims(img\_array, axis=0) # Add batch dimension

return img\_array

# Step 3: Prediction Function

def predict\_image(image):

processed\_image = preprocess\_image(image)

predictions = model.predict(processed\_image)

score = tf.nn.softmax(predictions[0])

return class\_names[np.argmax(score)], 100 \* np.max(score)

# Step 4: Streamlit UI

st.title("Flower Classification with MobileNetV2")

st.write("Upload an image of a flower, and the app will predict the type!")

uploaded\_file = st.file\_uploader("Choose an image...", type=["jpg", "jpeg", "png"])

if uploaded\_file is not None:

image = Image.open(uploaded\_file)

st.image(image, caption="Uploaded Image", use\_column\_width=True)

st.write("Classifying the image...")

label, confidence = predict\_image(image)

st.write(f"Prediction: \*\*{label}\*\*")

st.write(f"Confidence: \*\*{confidence:.2f}%\*\*")

**11. Conclusion**

In this project, we successfully built a deep learning model using MobileNetV2 to classify images of flowers into five different categories. The model was integrated into a Streamlit web application to provide a simple and interactive interface for real-time predictions. With high accuracy on the validation set and efficient real-time processing, the project demonstrates the power of transfer learning combined with user-friendly web interfaces.

**12. Future Work**

Model Optimization: Experiment with unfreezing the base MobileNetV2 model for fine-tuning to potentially improve accuracy.

Expand Dataset: Incorporate more flower types to make the model more versatile.

Deployment: Deploy the Streamlit app to a cloud platform like Heroku or Streamlit Sharing for broader accessibility.

This project demonstrates the effectiveness of using deep learning models like MobileNetV2 for image classification and shows how real-time predictions can be easily implemented using tools like Streamlit.