**Project Overview**

**Objective:**

Automatically identify **rotten vs. fresh fruits/vegetables** using a trained deep learning model with **transfer learning**.

**📦 Step 1: Data Preparation**

You need a dataset with labeled images of **fresh** and **rotten** fruits/vegetables. You can:

* Use Kaggle Datasets
* Build a custom dataset
* Augment to balance classes

Assume directory structure:

bash

CopyEdit

/dataset

/train

/fresh\_apples

/rotten\_apples

...

/val

/fresh\_apples

/rotten\_apples

**🔁 Step 2: Load & Augment Data**

python

CopyEdit

import tensorflow as tf

from tensorflow.keras.preprocessing.image import ImageDataGenerator

IMG\_SIZE = (224, 224)

BATCH\_SIZE = 32

train\_dir = 'dataset/train'

val\_dir = 'dataset/val'

train\_datagen = ImageDataGenerator(

rescale=1./255,

rotation\_range=30,

zoom\_range=0.2,

horizontal\_flip=True

)

val\_datagen = ImageDataGenerator(rescale=1./255)

train\_data = train\_datagen.flow\_from\_directory(

train\_dir,

target\_size=IMG\_SIZE,

batch\_size=BATCH\_SIZE,

class\_mode='categorical'

)

val\_data = val\_datagen.flow\_from\_directory(

val\_dir,

target\_size=IMG\_SIZE,

batch\_size=BATCH\_SIZE,

class\_mode='categorical'

)

**🔁 Step 3: Transfer Learning with MobileNetV2**

python

CopyEdit

from tensorflow.keras.applications import MobileNetV2

from tensorflow.keras import layers, models

base\_model = MobileNetV2(input\_shape=IMG\_SIZE + (3,), include\_top=False, weights='imagenet')

base\_model.trainable = False # Freeze the base

model = models.Sequential([

base\_model,

layers.GlobalAveragePooling2D(),

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

layers.Dropout(0.3),

layers.Dense(train\_data.num\_classes, activation='softmax')

])

model.compile(

optimizer='adam',

loss='categorical\_crossentropy',

metrics=['accuracy']

)

**🧪 Step 4: Train the Model**

python

CopyEdit

EPOCHS = 10

history = model.fit(

train\_data,

validation\_data=val\_data,

epochs=EPOCHS

)

**📈 Step 5: Evaluate & Fine-tune (Optional)**

You can unfreeze the base model for fine-tuning:

python

CopyEdit

base\_model.trainable = True

model.compile(

optimizer=tf.keras.optimizers.Adam(1e-5), # Lower LR

loss='categorical\_crossentropy',

metrics=['accuracy']

)

fine\_tune\_epochs = 5

model.fit(

train\_data,

validation\_data=val\_data,

epochs=fine\_tune\_epochs

)

**💾 Step 6: Save and Predict**

python

CopyEdit

model.save('smart\_sort\_model.h5')

# Load and predict

from tensorflow.keras.preprocessing import image

import numpy as np

img\_path = 'test\_image.jpg'

img = image.load\_img(img\_path, target\_size=IMG\_SIZE)

x = image.img\_to\_array(img)

x = np.expand\_dims(x, axis=0) / 255.

prediction = model.predict(x)

predicted\_class = np.argmax(prediction)

print(f"Predicted Class: {predicted\_class}")

**📊 Extras: Visualize Accuracy and Loss**

python

CopyEdit

import matplotlib.pyplot as plt

plt.plot(history.history['accuracy'], label='Train Accuracy')

plt.plot(history.history['val\_accuracy'], label='Val Accuracy')

plt.legend()

plt.title('Accuracy')

plt.show()