Project Title: Smart Sorting Transfer Learning for Identifying Rotten Fruits and Vegetables

Branch Name: Electronics and Communication Engineering

Track: Artificial Intelligence and Machine learning

Team Lead: Karapakula Balaji Sai Vrushadree

Mail Id : kbsv2004@gmail.com

Team Members:

- D Bhanu Parakash
- K Bhanu Prakash
- S. Bindusagar

Email Id's:

- Dugganibhanuprakash@gmail.com
- Bp2722123@gmail.com
- sbindusagar2003@gmail.com

Submitted By:

- Karapakula Balaji Sai Vrushadree, S. Bindusagar
- Annamacharya Institute of Technology & Sciences
- Electronics and Communication Engineering
- Roll No: 22AK1A0406, 22AK1A0412

Submitted To:

SmartBridge

Abstract:

This project introduces **Smart Sorting**, a computer vision solution that utilizes Transfer Learning to detect and classify spoiled fruits and vegetables. The main goal is to streamline the quality control process in the agricultural sector by minimizing manual errors and enhancing operational efficiency. The system is built using a Convolutional Neural Network (CNN) architecture based on the VGG16 model.

1. Introduction:

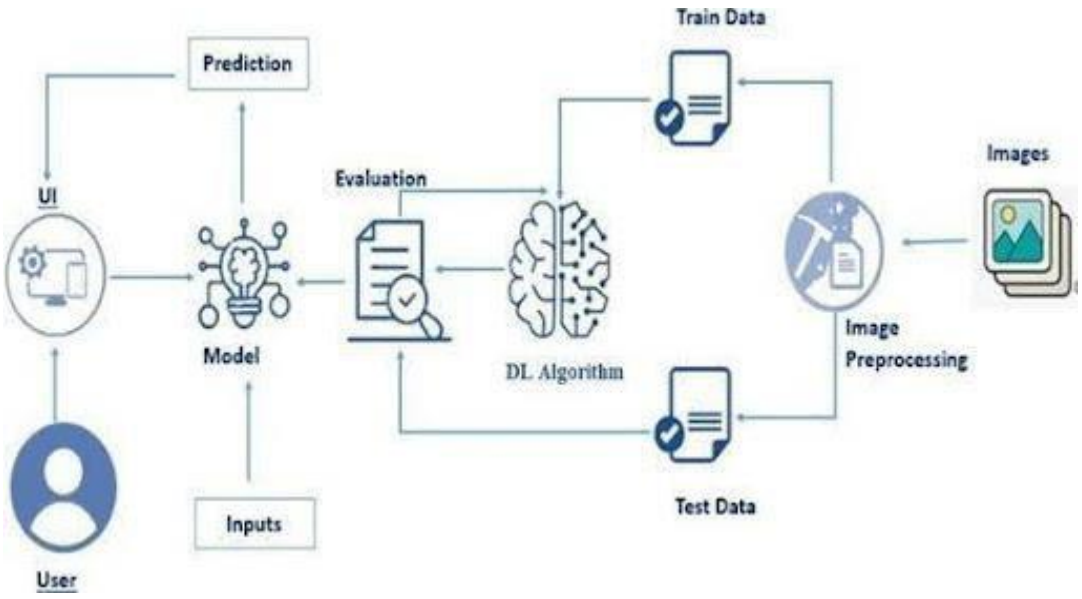
Sorting perishable items such as fruits and vegetables by hand is often slow, inconsistent, and error-prone. This project proposes an automated approach using deep learning techniques, specifically transfer learning with pre-trained models, to deliver a more accurate, scalable, and cost-efficient solution for quality assessment in the agricultural sector.

2. Problem Statement:

Quality control within the food supply chain presents notable challenges. This project aims to tackle issues :

- The inefficiency and inconsistency of manual sorting methods
- Increased waste resulting from delayed identification of spoilage
- The demand for real-time, on-the-spot sorting solutions in households and local markets

ARCHITECTURE:



PREREQUISITES :

- To complete this project, you must require the following software, concepts, and packages
  - Anaconda Navigator:
    - Refer to the link below to download Anaconda Navigator
  - Python packages:
  - Open anaconda prompt as administrator
  - Type “pip install numpy” and click enter.
  - Type “pip install pandas” and click enter.
  - Type “pip install scikit-learn” and click enter.
  - Type ”pip install matplotlib” and click enter.
  - Type ”pip install scipy” and click enter.
  - Type ”pip install seaborn” and click enter.
  - Type ”pip install tenser flow” and click enter.
  - Type “pip install Flask” and click enter.

A) PRIOR KNOWLEDGE

- DL Concepts
  - Neural Networks:: https://www.analyticsvidhya.com/blog/2020/02/cnn-vs-rnn-vs-mlp-analyzing-3-types-of-neural-networks-in-deep-learning/
  - Deep Learning Frameworks:: https://www.knowledgehut.com/blog/data-science/pytorch-vs-tensorflow
  - Transfer Learning: https://towardsdatascience.com/a-demonstration-of-transfer-learning-of-vgg-convolutional-neural-network-pre-trained-model-with-c9f5b8b1ab0a

- VGG16: https://www.geeksforgeeks.org/vgg-16-cnn-model/
- Convolutional Neural Networks (CNNs): https://www.analyticsvidhya.com/blog/2021/05/convolutional-neural-networks-cnn/ s://www.javatpoint.com/k-nearest-neighbor-algorithm-for-machine-learning
- Overfitting and Regularization: https://www.analyticsvidhya.com/blog/2021/07/prevent-overfitting-using-regularization-techniques/
- Optimizers: https://www.analyticsvidhya.com/blog/2021/10/a-comprehensive-guide-on-deep-learning-optimizers/

- Flask Basics: https://www.youtube.com/watch?v=lj4I_CvBnt0

B) PROJECT OBJECTIVES

By the end of this project, you will:

- Know fundamental concepts and techniques used for Deep Learning.
- Gain a broad understanding of data.
- Have knowledge of pre-processing the data/transformation techniques on outliers and some visualization concepts.

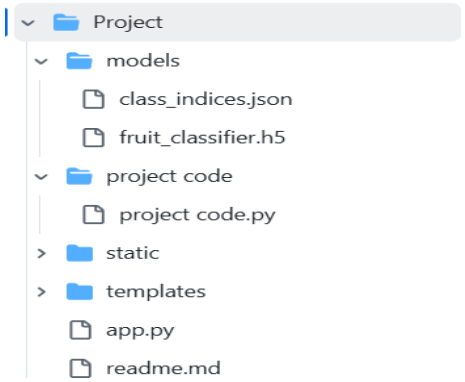
C) PROJECT FLOW

- The user interacts with the UI (User Interface) to choose the image.
- The chosen image is analysed by the model which is integrated with the flask application.
- Once the model analyses the input the prediction is showcased on the UI

- To accomplish this, we have to complete all the activities listed below,
- Data Collection: Collect or download the dataset that you want to train.
- Data pre-processing
  - Data Augmentation
  - Splitting data into train and test
- Model building
  - Import the model-building libraries
  - Initializing the model
  - Training and testing the model
  - Evaluating the performance of the model
  - Save the model
- Application Building
  - Create an HTML file
  - Build python code

Project Structure

Create the Project folder which contains files as shown below



- We are building a Flask application with HTML pages stored in the templates folder
- We are building a Python script app.py for scripting.
- fruit_classifier.h5 is our saved model. Further, we will use this model for flask integration.

DATA COLLECTION AND PREPARATION

Machine Learning relies significantly on data, as it forms the foundation for training effective algorithms. In this section, users can access and download the necessary dataset required for model development and evaluation.

A) COLLECTING THE DATASET

Data plays a central role in machine learning—it is the key ingredient that enables effective model training. This section guides you through the process of downloading the required dataset for this project.

Activity 1: Download the dataset

There are several open-source platforms available for accessing datasets, such as Kaggle, the UCI Machine Learning Repository, and others. For this project, we utilize a dataset consisting of 28 classes of fruits and vegetables, which can be downloaded directly from Kaggle or accessed via the Kaggle API.
Link: Dataset

Once the dataset is downloaded, the next step is to read, explore, and understand the data. This will be done using basic visualization and analysis techniques to uncover patterns and insights.

B) Data Visualization

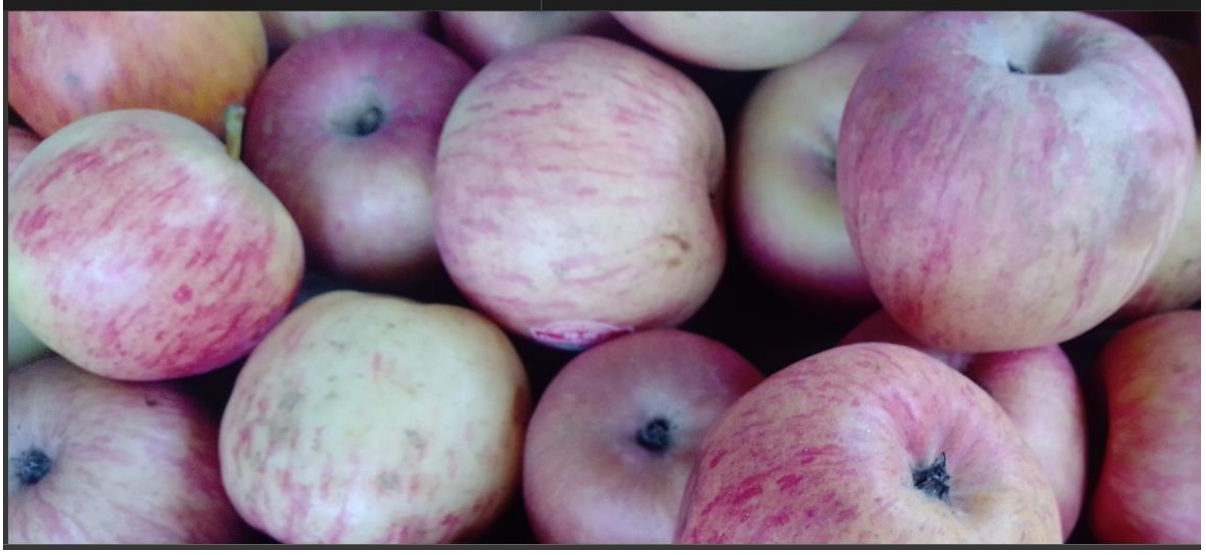
The given Python code begins by importing essential libraries for image handling and manipulation. It then selects a random image file from a designated directory and displays it using the Image module from python. This functionality is particularly useful for tasks such as exploring image datasets or testing image processing workflows by visualizing random samples from the data.

```
# Specify the path to your image folder
folder_path = '/content/output_dataset/train/Apple__Healthy'  # Replace with the actual path to your image folder

# List all files in the folder
image_files = [f for f in os.listdir(folder_path) if f.endswith(('.jpg', '.png', '.jpeg'))]

# Select a random image from the list
selected_image = random.choice(image_files)

# Display the randomly selected image
image_path = os.path.join(folder_path, selected_image)
display(Image(filename=image_path))
```



C) DATA AUGMENTATION

Data augmentation is a widely used technique in machine learning, especially in computer vision tasks like image classification. In projects such as distinguishing between healthy and rotten fruits and vegetables, augmentation helps improve model performance by artificially expanding the training dataset. This is achieved by applying various transformations—such as rotation, flipping, scaling, and adjustments in brightness or contrast—to existing images. These modifications enhance the diversity of the data and help the model become more robust to real-world variations.

For the 28-class classification task in this project, data augmentation typically boosts the model's ability to generalize by simulating the different ways objects may appear in actual scenarios. However, in this specific implementation, the dataset used was already preprocessed and cropped from an augmented dataset. Therefore, data augmentation was intentionally skipped. While this slightly increased the training time, it had minimal impact on the model's accuracy.

SPLIT DATA AND MODEL BUILDING

Train-Test Split :

In this project, the dataset has been pre-divided into separate training and testing sets, eliminating the need for manual data splitting.

```
trainpath = "/content/output_dataset/train"
testpath= "/content/output_dataset/test"

train_datagen = ImageDataGenerator(rescale = 1./255,zoom_range= 0.2,shear_range= 0.2)
test_datagen = ImageDataGenerator(rescale = 1./255)

train = train_datagen.flow_from_directory(trainpath,target_size =(224,224),batch_size = 20)
test = test_datagen.flow_from_directory(testpath,target_size =(224,224),batch_size = 20) ,#5 ,15 , 32, 50

Found 3358 images belonging to 28 classes.
Found 1120 images belonging to 28 classes.
```

A) Model Building:

VGG16 Transfer Learning Model

The model in this project leverages the pre-trained VGG16 architecture with its convolutional base layers frozen to retain previously learned features. Built using a sequential structure, the model includes the VGG16 base, followed by a flattening layer, a dropout layer for regularization, and a dense output layer with SoftMax activation to classify images into five categories.

The model is compiled with the Adam optimizer and sparse categorical cross-entropy as the loss function. Training is carried out over 10 epochs using a data generator for the training set. To improve performance and avoid overfitting, Model Checkpoint and Early Stopping callbacks are implemented. The best version of the trained model is saved as fruit_classifier.h5 for future use.

A model summary is provided post-compilation, displaying the complete architecture, including the number of layers and trainable parameters.

```
from tensorflow.keras.applications.vgg16 import VGG16
from tensorflow.keras.layers import Dense, Flatten
from tensorflow.keras.models import Model
```

```
vgg = VGG16(include_top = False,input_shape = (224,224,3))
```

```
for layer in vgg.layers:
  print(layer)
```

```
<keras.src.engine.input_layer.InputLayer object at 0x79c096fde230>
<keras.src.layers.convolutional.conv2d.Conv2D object at 0x79c096fde4d0>
<keras.src.layers.convolutional.conv2d.Conv2D object at 0x79c0081b7a90>
<keras.src.layers.pooling.max_pooling2d.MaxPooling2D object at 0x79bff7ef2f80>
<keras.src.layers.convolutional.conv2d.Conv2D object at 0x79c094405810>
<keras.src.layers.convolutional.conv2d.Conv2D object at 0x79c00834ba30>
<keras.src.layers.pooling.max_pooling2d.MaxPooling2D object at 0x79bff6dad540>
<keras.src.layers.convolutional.conv2d.Conv2D object at 0x79c096fd2c20>
<keras.src.layers.convolutional.conv2d.Conv2D object at 0x79c094405360>
<keras.src.layers.convolutional.conv2d.Conv2D object at 0x79c094405db0>
<keras.src.layers.pooling.max_pooling2d.MaxPooling2D object at 0x79bfcc0fc490>
<keras.src.layers.convolutional.conv2d.Conv2D object at 0x79bff6dae7d0>
<keras.src.layers.convolutional.conv2d.Conv2D object at 0x79bff6dad4b0>
<keras.src.layers.convolutional.conv2d.Conv2D object at 0x79bff6dae020>
<keras.src.layers.pooling.max_pooling2d.MaxPooling2D object at 0x79bfcc0fff10>
<keras.src.layers.convolutional.conv2d.Conv2D object at 0x79bfcc0fe0b0>
<keras.src.layers.convolutional.conv2d.Conv2D object at 0x79bfcc0fe770>
<keras.src.layers.convolutional.conv2d.Conv2D object at 0x79bfcc0fd300>
<keras.src.layers.pooling.max_pooling2d.MaxPooling2D object at 0x79bfcc0fe650>
```

```
len(vgg.layers)
```

```
19
```

```
for layer in vgg.layers:
  layer.trainable = False
```

```
x= Flatten()(vgg.output)
```

```
output = Dense(28, activation ='softmax')(x)
```

```
vgg16 = Model(vgg.input,output)
```

```
vgg16.summary()
```

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 224, 224, 3)]     0

 block1_conv1 (Conv2D)       (None, 224, 224, 64)      1792

 block1_conv2 (Conv2D)       (None, 224, 224, 64)      36928

 block1_pool (MaxPooling2D)  (None, 112, 112, 64)      0

 block2_conv1 (Conv2D)       (None, 112, 112, 128)     73856

 block2_conv2 (Conv2D)       (None, 112, 112, 128)     147584

 block2_pool (MaxPooling2D)  (None, 56, 56, 128)       0

 block3_conv1 (Conv2D)       (None, 56, 56, 256)       295168

 block3_conv2 (Conv2D)       (None, 56, 56, 256)       590080

 block3_conv3 (Conv2D)       (None, 56, 56, 256)       590080
```

TESTING MODEL & DATA PREDICTION

The trained **VGG16 model** is evaluated using the predict() function to generate predictions on new or unseen data. This step helps verify the model's performance and its ability to correctly classify images based on the learned features.

```
labels=[0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27]
```

Testing class - 1

```
img_path = '/content/output_dataset/train/Bellpepper__Healthy/freshPepper (104).jpg'
```

```
import numpy as np
img = load_img(img_path, target_size=(224, 224))
x = img_to_array(img)
x = preprocess_input(x)
preds = vgg16.predict(np.array([x]))
preds
```

```
1/1 [==============================] - 0s 128ms/step
array([[0., 0., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
        0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.]], dtype=float32)
```

```
labels[np.argmax(preds)]
```

```
4
```

Testing class-2

```
img_path = '/content/output_dataset/train/Mango__Rotten/153.jpg'
```

```
import numpy as np
img = load_img(img_path, target_size=(224, 224))
x = img_to_array(img)
x = preprocess_input(x)
preds = vgg16.predict(np.array([x]))
preds
```

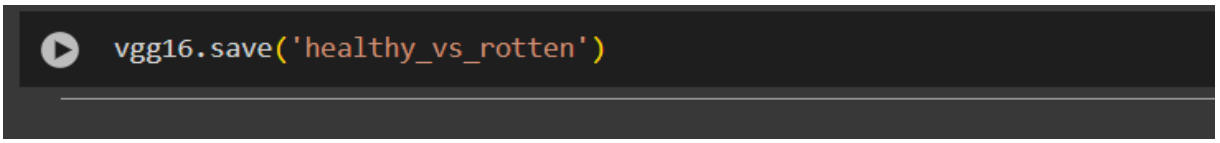
```
1/1 [==============================] - 0s 19ms/step
array([[0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
        0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.]], dtype=float32)
```

```
labels[np.argmax(preds)]
```

```
17
```

SAVING THE MODEL

Finally, we have chosen the best model now saving that model



Objective:

- Build a Machine Learning Model to classify fruits and vegetables as healthy or rotten using image data.
- Develop a Web-Based Application for users to easily upload and classify produce images in real-time.
- Minimize Food Waste and Human Effort by automating the quality control process and reducing manual inspection.

Scenarios of Application:

- Food Processing Plants
- Supermarkets
- Smart Home

Tools and Technologies Used:

- Anaconda, Python, Flask
- Libraries: NumPy, Pandas, Scikit-learn, TensorFlow, Matplotlib

Concepts and Prerequisites:

- Deep Learning & CNN
- Transfer Learning using VGG16
- Flask for web deployment

Methodology:

- Dataset: 28-class fruit/vegetable images from Kaggle

- Preprocessing: Normalization, Augmentation

- Model: VGG16 with SoftMax, trained over 10 epochs

- Accuracy validated via prediction samples

System Architecture:

- UI Input
- Flask API
- Model Prediction
- Output Displayed

Application Building

In this section, we develop a web application integrated with the trained machine learning model. A user-friendly interface (UI) is provided, allowing users to upload images for prediction. The input is passed to the saved model, and the classification result is displayed directly on the interface.

Key Steps Involved:

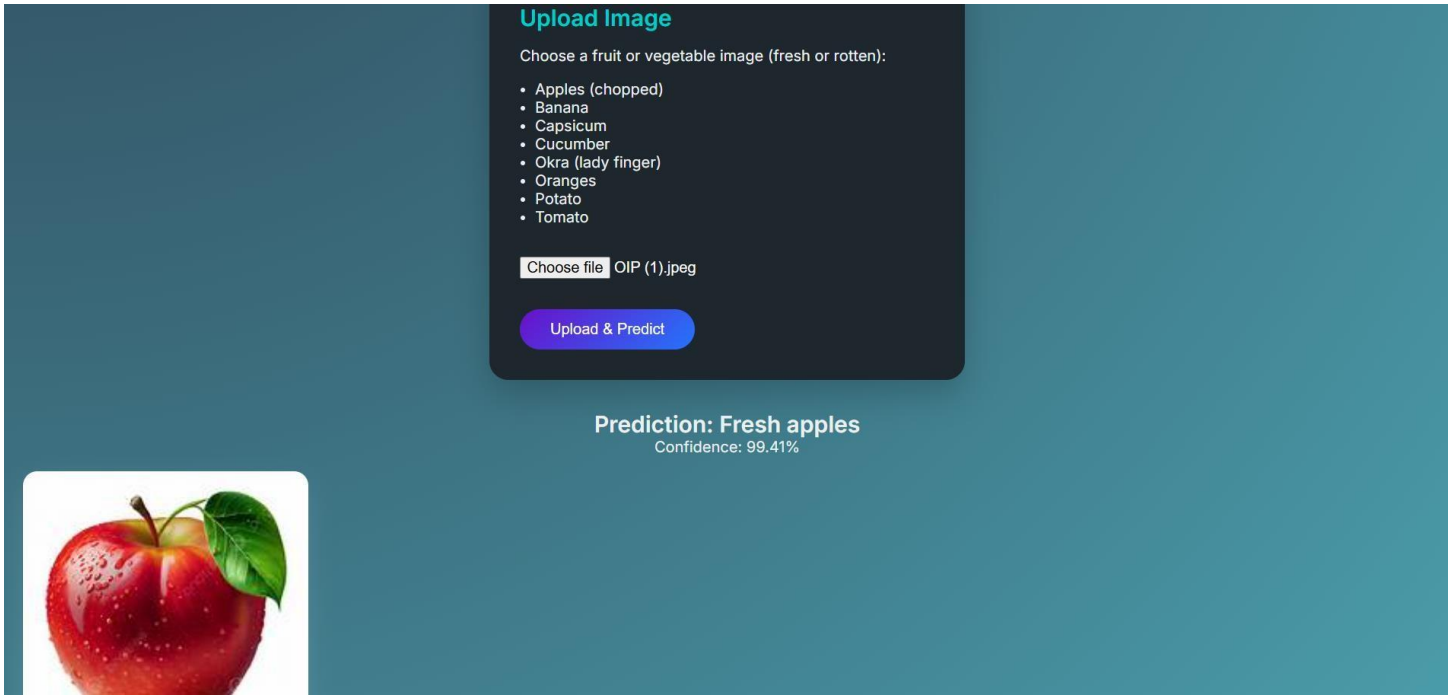
- Creating HTML Pages: Designing the front-end interface for image upload and result display.
- Developing Server-Side Script: Implementing the backend logic to handle image input.

Results and Observations:

The model accurately classified various test samples such as:

- Rotten Apple
- Healthy Apple

It supports classification across 28 categories of fruits and vegetables, including examples like Apples (chopped), Banana, Capsicum, Cucumber, Okra (Lady Finger), Oranges, Potato, and Tomato—in both fresh and rotten conditions. The predictions were consistently accurate, showcasing the model's ability to generalize across diverse inputs and confirming its effectiveness for real-time freshness detection.



2. Conclusion:

This project demonstrates how transfer learning can enable efficient image-based classification and automation in the agricultural sector, significantly reducing both food waste and reliance on manual labor.

3. Future Scope:

- Broader spoilage detection
- IoT integration
- Mobile app support

4. References :

- Kaggle dataset
- Analytics Vidhya
- Geeks for Geeks VGG16