Identifying Rotten Fruits and Vegetables Using Transfer Learning

Final Report

Date	31 January 2025
Team ID	LTVIP2026TMID91218
Project Name	Transfer learning for identifying Rotten fruits and vegetables
Maximum Marks	4 Marks

Team ID : LTVIP2026TMIDS91218

Team Size : 4

Team Leader : Baba Vali Shaik(project helper)

Team member : Konreddy Vani

Team member : Mallepogula Ganesh(project head)

Team member : Keerthi Baddela

1. INTRODUCTION

1.1 Project Overview

Food quality inspection is a critical process in agriculture and food supply chains. Manual inspection of fruits and vegetables is time-consuming, error-prone, and inconsistent. This project proposes an AI-based system using Transfer Learning to automatically identify rotten and fresh fruits and vegetables from images. The system leverages pre-trained deep learning models to achieve high accuracy with limited data and reduced training time.

1.2 Purpose

The purpose of this project is to:

- Reduce food wastage
- Improve quality control
- Automate spoilage detection
- Provide a fast, accurate, and scalable solution using deep learning

---

2. IDEATION PHASE

2.1 Problem Statement

Manual inspection methods fail to detect early-stage spoilage in fruits and vegetables, leading to financial loss, food wastage, and inconsistent quality. There is a need for an automated, image-based solution to accurately classify produce as fresh or rotten**.**

2.2 Empathy Map Canvas

- Users: Farmers, vendors, warehouse inspectors
- Pain Points: Human error, time consumption, wastage
- Needs: Accuracy, speed, automation
- Emotions: Frustration before, confidence after solution

2.3 Brainstorming

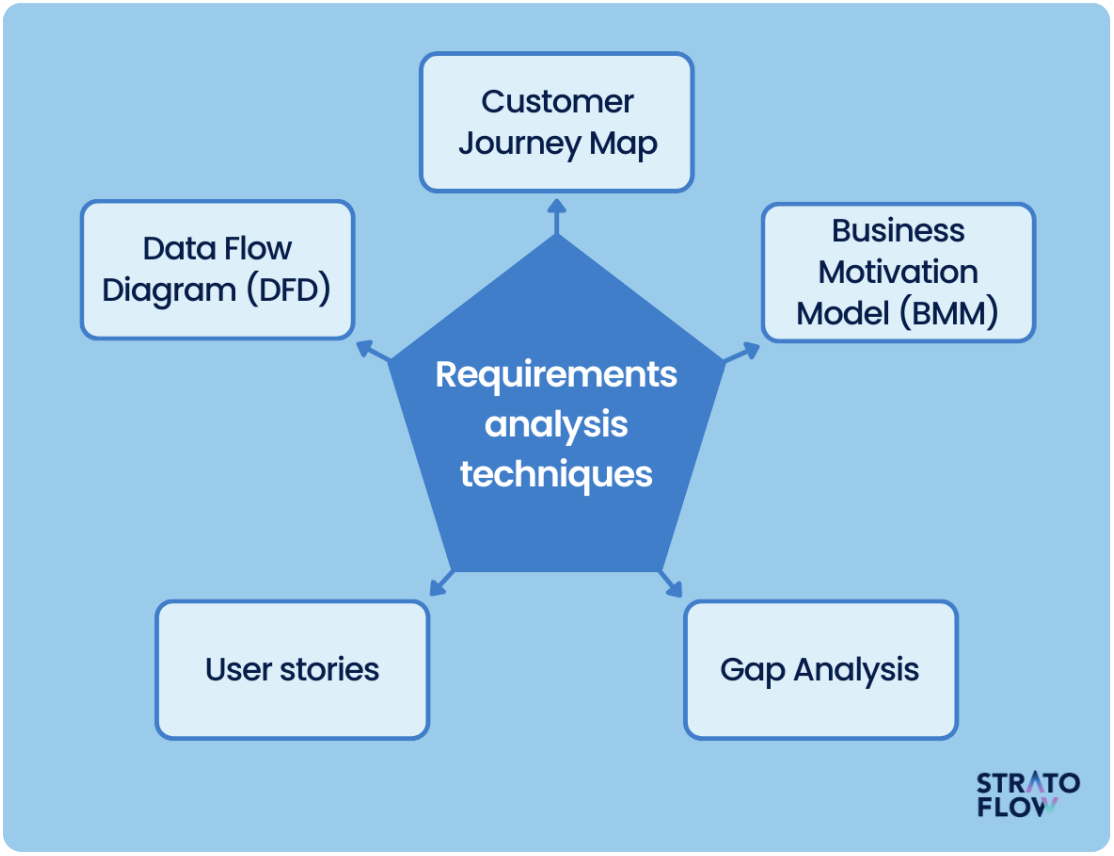
Multiple ideas were discussed such as traditional image processing, ML classifiers, and deep learning models. After evaluation, transfer learning with CNNs was selected due to its accuracy and efficiency.

---

3. REQUIREMENT ANALYSIS

3.1 Customer Journey Map

1. Capture or upload fruit image
2. System processes image
3. AI model classifies produce
4. Result displayed to user
5. Decision made (use/discard)



3.2 Solution Requirement

Functional Requirements

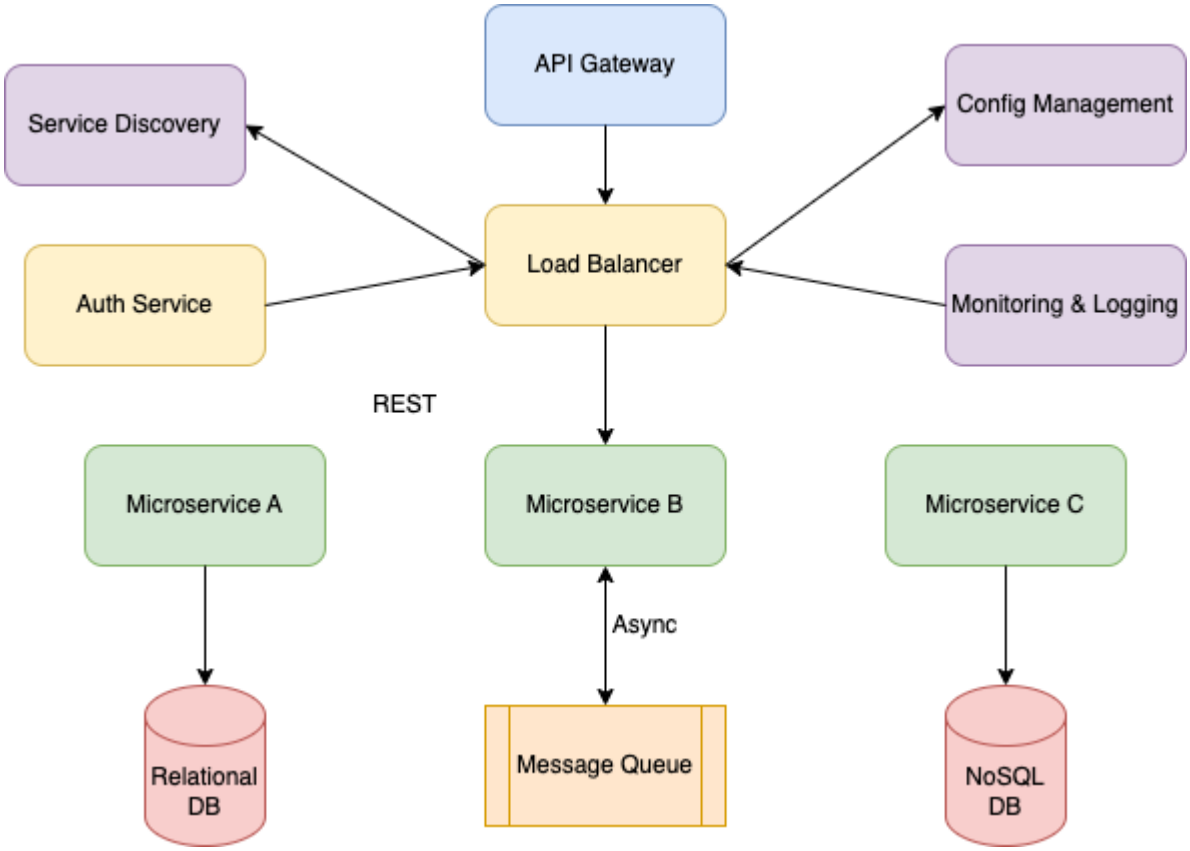
- User registration & login
- Image upload/capture

- Classification (Fresh/Rotten)
- Result display with confidence score

Non-Functional Requirements

- High accuracy
- Fast response time
- Scalability
- **Security & reliability**

3.3 Data Flow Diagram



3.4 Technology Stack

Layer	Technology
Frontend	HTML, CSS
Backend	Python
ML Framework	TensorFlow / Keras
Libraries	NumPy, OpenCV, Matplotlib
IDE	Jupyter Notebook

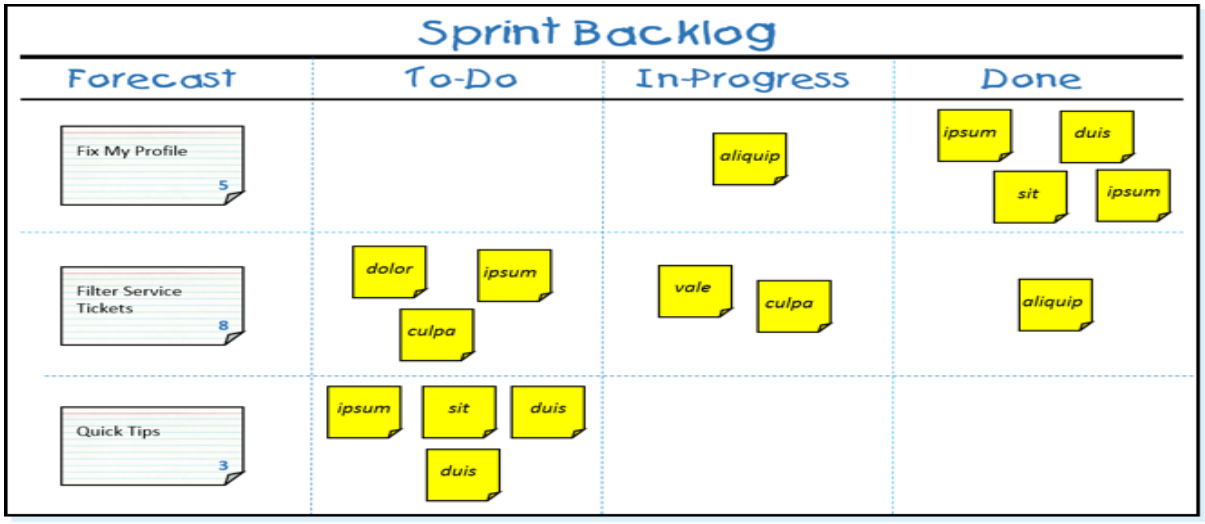
Layer	Technology
Deployment (Optional)	**Flask**

---

4. PROJECT DESIGN

4.1 Problem–Solution Fit

The solution directly automates an existing manual process using image capture, ensuring quick adoption, reduced errors, and effective spoilage detection**.**



4.2 Proposed Solution

An AI-based image classification system using pre-trained CNN models (VGG, ResNet, MobileNet) to identify rotten fruits and vegetables with high accuracy.

4.3 Solution Architecture

- User Interface
- Application Layer
- Image Preprocessing
- Transfer Learning Model
- Classification Layer
- Result Display & Storage

---

5. PROJECT PLANNING & SCHEDULING

5.1 Project Planning

Agile methodology with multiple sprints:

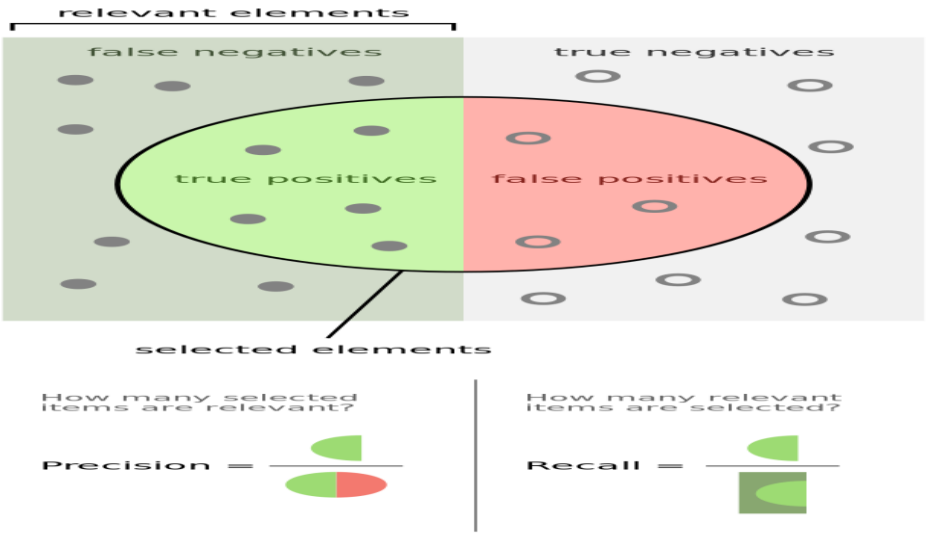
- Sprint 1: Data Collection & Preparation
- Sprint 2: Model Training
- Sprint 3: Evaluation & Optimization
- Sprint 4: Integration & Deployment

---

6. FUNCTIONAL AND PERFORMANCE TESTING

6.1 Performance Testing

- Model accuracy testing
- Response time evaluation
- Testing on unseen images
- Validation against overfitting



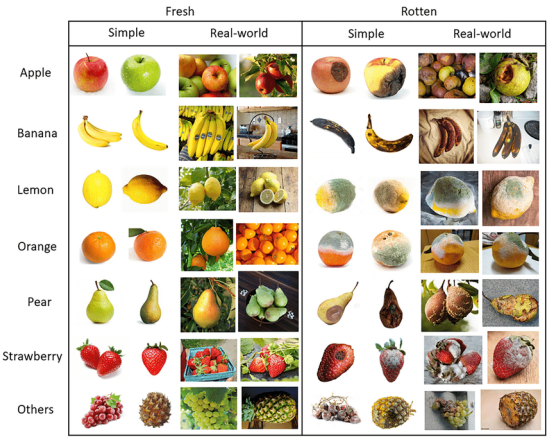
---

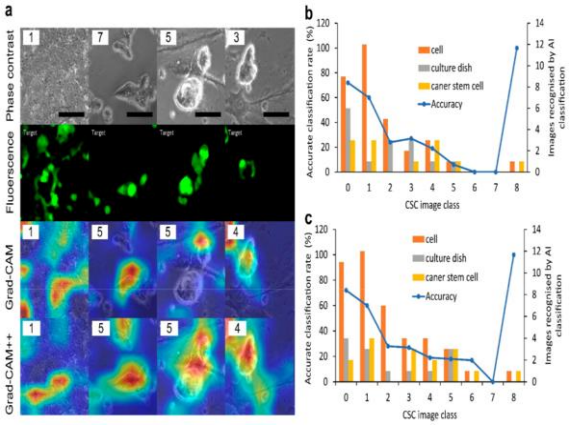
7. RESULTS

7.1 Output Screenshots

- Image upload screen
- Classification output (Fresh / Rotten)
- Confidence score display

(Screenshots to be added in final document)





---

8. ADVANTAGES & DISADVANTAGES

Advantages

- Automated detection
- High accuracy
- Reduces food wastage
- Time-efficient
- Scalable solution

Disadvantages

- Requires quality images
- Model retraining needed for new produce types
- **Initial setup cost**

---

9. CONCLUSION

The project successfully demonstrates the use of transfer learning to solve a real-world agricultural problem. The system provides accurate, fast, and reliable identification of rotten fruits and vegetables, reducing manual effort and improving quality control.

---

10. FUTURE SCOPE

- Mobile application deployment
- Real-time camera integration
- Multi-class fruit/vegetable identification
- Cloud-based scalable deployment
- IoT integration for smart farming

---

11. APPENDIX

Source Code

```
import cv2
import numpy as np
from roboflow import Roboflow
from flask import Flask, render_template, request, Response, redirect, url_for
import os
import uuid  # Add this to generate unique filenames

# Initialize Roboflow with your API key (use your own API key)
rf = Roboflow(api_key="RBMCiagFraeIHPvptwcS")
project = rf.workspace().project("freshness-fruits-and-vegetables")
model = project.version(7).model

# Initialize Flask app
app = Flask(__name__)
UPLOAD_FOLDER = 'static/uploads/'
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER

# Helper function for object detection on an image
def detect_on_image(image_path):
    image = cv2.imread(image_path)
    results = model.predict(image, confidence=40, overlap=30).json()

    for prediction in results['predictions']:
        x, y, w, h = (
            prediction['x'],
            prediction['y'],
            prediction['width'],
            prediction['height']
        )
        class_name = prediction['class']
```

```
            confidence = prediction['confidence']

            # Calculate bounding box coordinates
            x_min = int(x - w / 2)
            y_min = int(y - h / 2)
            x_max = int(x + w / 2)
            y_max = int(y + h / 2)

            # Draw the bounding box
            cv2.rectangle(image, (x_min, y_min), (x_max, y_max), (0, 255, 0), 2)

            # Create label with class and confidence score
            label = f"{class_name}: {confidence:.2f}"

            # Put the label above the bounding box
            cv2.putText(image, label, (x_min, y_min - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255, 0), 2)

    # Save the result image with a unique filename
    result_filename = f'result_{uuid.uuid4().hex}.jpg'
    result_image_path = os.path.join(UPLOAD_FOLDER, result_filename)
    cv2.imwrite(result_image_path, image)

    return result_image_path

# Route for homepage with detection options
@app.route('/')
def index():
    return render_template('index.html')

# Route for handling image upload and detection
@app.route('/detect_image', methods=['POST'])
def detect_image():
```

```
    if 'image' not in request.files:
        return redirect(request.url)

    file = request.files['image']

    if file.filename == '':
        return redirect(request.url)

    if file:
        # Save the uploaded image
        image_path = os.path.join(app.config['UPLOAD_FOLDER'], file.filename)
        file.save(image_path)

        # Perform detection on the image
        result_image_path = detect_on_image(image_path)
        result_image_filename = os.path.basename(result_image_path)  # Get the filename only
        os.remove(image_path)  # Optionally remove the uploaded image after processing

        # Generate a UUID for cache-busting
        unique_id = uuid.uuid4().hex

        # Render the result page with the processed image and UUID
        return render_template('result.html', result_image=result_image_filename,
unique_id=unique_id)

# Route for handling live webcam feed detection
@app.route('/live_feed')
def live_feed():
    return render_template('live_feed.html')

def generate_live_feed():
    cap = cv2.VideoCapture(0)
```

```
if not cap.isOpened():
    raise IOError("Cannot open webcam")

while True:
    ret, frame = cap.read()
    if not ret:
        break

    # Resize the frame to 640x640
    frame = cv2.resize(frame, (640, 640))

    # Perform inference on each resized frame
    results = model.predict(frame, confidence=40, overlap=30).json()

    # Extract bounding boxes and labels from the predictions
    for prediction in results['predictions']:
        x, y, w, h = (
            prediction['x'],
            prediction['y'],
            prediction['width'],
            prediction['height']
        )
        class_name = prediction['class']
        confidence = prediction['confidence']

        # Calculate bounding box coordinates
        x_min = int(x - w / 2)
        y_min = int(y - h / 2)
        x_max = int(x + w / 2)
        y_max = int(y + w / 2)

        # Draw the bounding box
```

```
            cv2.rectangle(frame, (x_min, y_min), (x_max, y_max), (0, 255, 0), 2)

            # Create label with class and confidence score
            label = f"{class_name}: {confidence:.2f}"

            # Put the label above the bounding box
            cv2.putText(frame, label, (x_min, y_min - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255, 0),
2)

        # Encode the frame into JPEG format
        ret, buffer = cv2.imencode('.jpg', frame)
        frame = buffer.tobytes()

        # Yield the frame in byte format for streaming
        yield (b'--frame\r\n'
               b'Content-Type: image/jpeg\r\n\r\n' + frame + b'\r\n')

    cap.release()

@app.route('/video_feed')
def video_feed():
    return Response(generate_live_feed(),
                    mimetype='multipart/x-mixed-replace; boundary=frame')

# Main entry point
if __name__ == '__main__':
    if not os.path.exists(UPLOAD_FOLDER):
        os.makedirs(UPLOAD_FOLDER)
    app.run(debug=True)
```

Dataset Link

https://www.kaggle.com/datasets/moltean/fruits

GitHub & Project Demo Link

https://github.com/GANESH10RR/Transfer-learning-for-identifying-Rotten-Fruit-and-vegetables-.git