2024

Rotton Fruit Detection

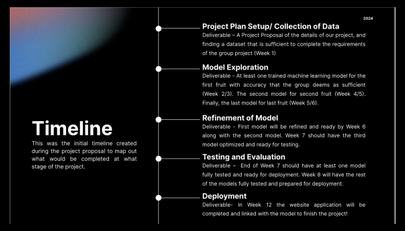
By: Richard Angueira, Michael Banko, Christian Rasmussen, & Brian Weideli

Table Of Contents

Introduction
Meet the Team, About our Project, etc.

Project
Our Idea, Purpose/End Goal, Project Features

Project (cont)
Timeline, Models



Code
Run_Model.py, Code Libraries, etc.

Meet the Team!

2024

Michael Banko - Project Leader/Developer

Michael is a senior at Kean University going for his Bachelor's in Computer Science. He has his associate's in Computer Science from Raritan Valley Community College. He enjoys hanging out with friends, playing video games, and listening to music

Richard Angueira - Developer

In a few weeks Richard will be graduating with a Bachelor's in Computer Science. He's all set to jump into work with his skills sharp and his mind focused. He is eager to start making a difference right away! His interests are photography, video games, and sleeping.

Christian Rasmussen - Developer/Scribe

Christian is a senior at Kean University. He is studying Computer Science and is graduating in May 2024. He is looking to get into the field of software engineering. He enjoys music, going to the gym, and hanging out with friends.

Brian Weideli - Developer

Brian is enrolled in the Computer Science program at Kean university. He is expected to graduate in May 2024. Once graduated, he would like to work as a data analyst.

Our Idea

Our idea was to create a software that would allow people to be able to upload an image of a piece of fruit and have it tell you whether that fruit was fresh or rotten based off of trained machine learning models.

Purpose

The purpose of our project is to allow both producers (farmers) and consumers (customers) to determine whether a fruit is either fresh for harvest or whether a fruit is good for consumption.

End Goal

The end result from this project is a website application that allows users to upload their own images of peaches, pomegranates, and strawberries. Once the website application receives the image, it then sends it to one of our machine learning models for classification. Once completed, the results get sent back to the website application for the user to see.

Project Features

Image Recognition: The project utilizes advanced image recognition algorithms to analyze visual features of fruits, distinguishing between fresh and rotten ones based on color, texture, and other visual characteristics.

Machine Learning Models: Machine learning models, particularly deep learning architectures like convolutional neural networks (CNNs), are trained on large datasets of labeled fruit images to accurately classify fruit freshness.

Real-Time Processing: The system provides real-time processing capabilities, allowing users to capture images of fruits using cameras or smartphones and receive instant feedback on their freshness status.

Customization and Adaptability: The classifier can be customized and adapted to different types of fruits and varying environmental conditions, ensuring robust performance across diverse scenarios and fruit varieties.

2024

Timeline

This was the initial timeline created during the project proposal to map out what would be completed at what stage of the project.

Project Plan Setup/ Collection of Data

Deliverable – A Project Proposal of the details of our project, and finding a dataset that is sufficient to complete the requirements of the group project (Week 1)

Model Exploration

Deliverable - At least one trained machine learning model for the first fruit with accuracy that the group deems as sufficient (Week 2/3). The second model for second fruit (Week 4/5). Finally, the last model for last fruit (Week 5/6).

Refinement of Model

Deliverable - First model will be refined and ready by Week 6 along with the second model. Week 7 should have the third model optimized and ready for testing.

Testing and Evaluation

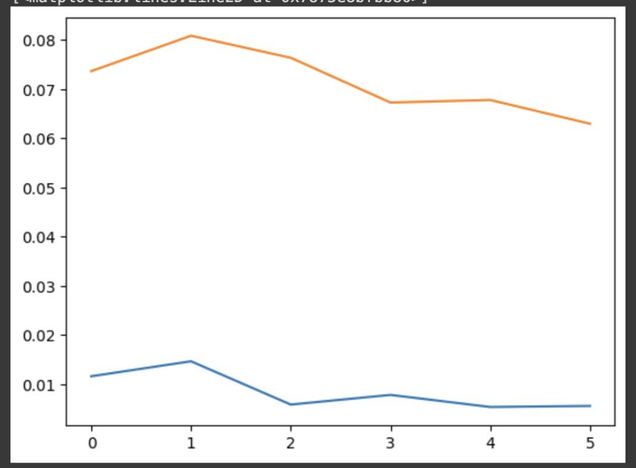
Deliverable – End of Week 7 should have at least one model fully tested and ready for deployment. Week 8 will have the rest of the models fully tested and prepared for deployment.

Deployment

Deliverable- In Week 12 the website application will be completed and linked with the model to finish the project!

Peach

```
Train loss: 0.011575429358830054
Test loss: 0.07369004841893911
Train loss: 0.014626355996976296
Test loss: 0.08090198365971446
Train loss: 0.005813967250287533
Test loss: 0.07641568267717957
Train loss: 0.00777896932947139
Test loss: 0.0672982637770474
Train loss: 0.005322869929174582
Test loss: 0.06782146589830518
Train loss: 0.005535525890688102
Test loss: 0.0629882479552179 6
[<matplotlib.lines.Line2D at 0x7875e8bfbb80>]
```

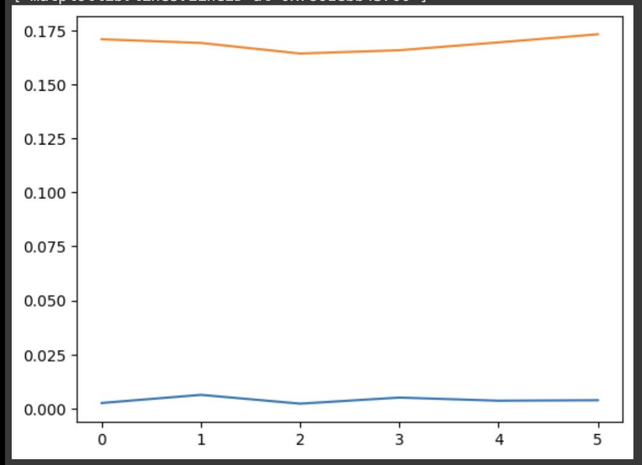



```
# Calculate accuracy
accuracy = accuracy_score(all_labels, all_preds)
print(f"Accuracy: {accuracy}")
```

```
Accuracy: 0.9663865546218487
```

Strawberry

```
Train loss: 0.0025419411876310524
Test loss: 0.17081166617572308
Train loss: 0.006308511526395495
Test loss: 0.16909842379391193
Train loss: 0.0022396712402741495
Test loss: 0.16420925129204988
Train loss: 0.005047398368613078
Test loss: 0.16572663933038712
Train loss: 0.0035980151375182546
Test loss: 0.1693662852048874
Train loss: 0.003824322233692958
Test loss: 0.17311611399054527
[<matplotlib.lines.Line2D at 0x7861ebb43700>]
```

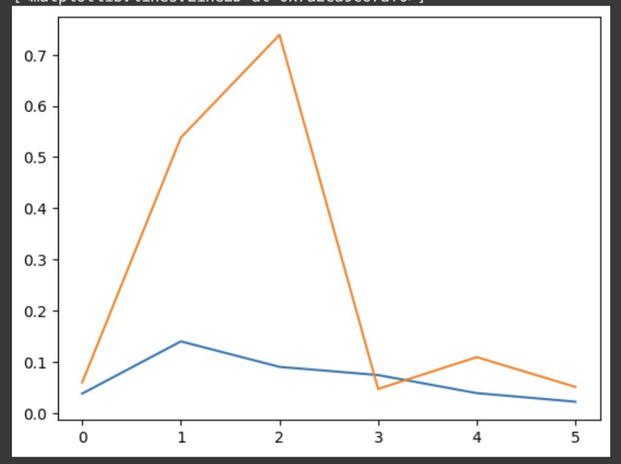



```
# Calculate accuracy
accuracy = accuracy_score(all_labels, all_preds)
print(f"Accuracy: {accuracy}")
```

```
Accuracy: 0.94
```

Pomegranate

```
Train loss: 0.037726328366746505
Test loss: 0.059647081419825554
Train loss: 0.13974593256910642
Test loss: 0.5380290485773003
Train loss: 0.08991213422268629
Test loss: 0.7390573954326101
Train loss: 0.07393523037123183
Test loss: 0.04680689424276352
Train loss: 0.03859610079477230 5
Test loss: 0.10895867682120297
Train loss: 0.02185118442090849
Test loss: 0.05080989096313715
[<matplotlib.lines.Line2D at 0x7a2ca9c67af0>]
```




```
# Calculate accuracy
accuracy = accuracy_score(all_labels, all_preds)
print(f"Accuracy: {accuracy}")
```

```
Accuracy: 0.972972972972973
```

Models

2024

Run_Model.py

```python
from flask import Flask, request, jsonify
import torch
import torchvision.transforms as transforms
from torchvision import models
from PIL import Image
import json
from flask import Flask, request, jsonify
from flask_cors import CORS

app = Flask(__name__)
CORS(app)

@app.route('/run_model', methods=['POST'])
def run_model():
    # Load the model architecture
    model = models.resnet18(pretrained=False)
    num_ftrs = model.fc.in_features
    model.fc = torch.nn.Linear(num_ftrs, 2)

    # Extract inputs from form data
    selected_fruit = request.form['fruitSelection']
    image_file = request.files['file']

    # Save image to a temporary location
    image_path = "temp_image.jpg"
    image_file.save(image_path)

    # Set model path based on selected fruit
    model_path = selected_fruit + ".pt"

    # Load the saved model weights
    try:
        model.load_state_dict(torch.load(model_path, map_location=torch.device('cpu')))
    except Exception as e:
        return jsonify({"error": "Error loading model"}), 500

    model.eval()
```

```python
    # Define the transform for the input image
    transform = transforms.Compose([
        transforms.Resize((300, 300)),
        transforms.ToTensor(),
        transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])
    ])

    # Load and preprocess the input image
    try:
        image = Image.open(image_path).convert("RGB")
        input_tensor = transform(image).unsqueeze(0)
    except Exception as e:
        return jsonify({"error": "Error loading image"}), 500

    # Classify the image
    with torch.no_grad():
        output = model(input_tensor)

    # Get the predicted class
    _, predicted_class = torch.max(output, 1)
    predicted_class = predicted_class.item()

    # Define class labels
    class_labels = ['Not Rotten', 'Rotten']

    # Adding disclaimer for if fruit is rotten
    result_message = class_labels[predicted_class]
    if result_message == 'Rotten':
        result_message += "Disclaimer: The classification provided by this software is intended as a recommendati


    # Send result back to client
    result = {
        "class": class_labels[predicted_class]
    }
    return jsonify(result)

if __name__ == '__main__':
    app.run(debug=True)  # You may want to set debug=False in production
```

Flask

Is used in the creation of web apps.

Torch

Is a tensor computation with GPU acceleration.

TorchVision

PIL(Python Imaging Library)

Used for image manipulation and processing.

Components of the code

Thank You!

Any questions?