A Comparative Analysis of Deep Learning and Handcrafted Feature-Based Approaches for Multiclass Fruit and Vegetable Classification

Doruk Dolaşık
Department of Artifical Intelligence Engineering
TOBB ETU
Email: ddolasik@etu.edu.tr

Abstract—This project addresses the challenge of detecting and classifying fresh and spoiled fruits and vegetables in the food supply chain. The primary objective is to develop an automated, non-contact, and high-accuracy visual analysis system that supports quality control, extends shelf life management, and reduces food waste. Two distinct datasets were used, both sourced from Kaggle: the Fruit and Vegetable Disease – Healthy vs Rotten dataset (21,044 images, 14 classes) and the Fresh and Stale Classification dataset (23,425 images, 28 classes).

In Method 1, a Convolutional Neural Network (CNN)–based deep learning model was implemented to perform end-to-end classification directly from raw images. Images were resized to 224×224 pixels, normalized to the [0,1] range, and split into training (60%), validation (20%), and test (20%) sets. In Method 2, a traditional machine learning approach was adopted, where all images were resized to 128×128 pixels, Histogram of Oriented Gradients (HOG) features were first extracted, followed by dimensionality reduction with PCA, then combined with Hue-Saturation-Value (HSV) color features, and finally classified using a Random Forest (RF) model.

Experimental results show that RF achieved the highest accuracy on both datasets, outperforming CNN by leveraging its ability to integrate both texture (HOG) and color (HSV) information in a feature-rich space. Statistical analysis revealed that RF maintained high precision and recall across all classes, with particularly strong performance in minority classes where CNN tended to overfit. The ensemble nature of RF, which aggregates predictions from multiple decorrelated decision trees, allowed it to handle intra-class variations more effectively, especially in cases with subtle spoilage indicators. While CNN demonstrated superior generalization to unseen variations and required no manual feature engineering, RF provided more stable accuracy under controlled conditions, shorter training times, and higher interpretability in feature importance analysis. This comparative study highlights the trade-offs between deep learning and feature-engineering-based approaches in terms of accuracy, generalization, interpretability, and computational efficiency.

Index Terms— Convolutional Neural Network, Image classification, HOG, HSV, Machine learning, Random Forest.

I. INTRODUCTION

A. *Problem Definition and Motivation*

In the modern food industry, ensuring product freshness is vital for maintaining quality, extending shelf life, and reducing food waste. Current inspection methods for determining the freshness or spoilage of fruits and vegetables are predominantly manual and often require physical contact. These traditional approaches are time-consuming, labor-intensive, and susceptible to human error, leading to inefficiencies in the supply chain. The SpoilScan project is motivated by the need for a fully automated, non-contact, and reliable inspection system that can quickly and accurately identify the condition of produce, enabling better decision-making in storage, distribution, and retail environments.

B. *Classification Approach*

The problem addressed in this project is a multi-class image classification task. Two alternative artificial intelligence approaches are explored:

Method 1: Deep learning using a Convolutional Neural Network (CNN) trained directly on raw images.

Method 2: Traditional machine learning using a feature-engineering pipeline consisting of Histogram of Oriented Gradients (HOG) feature extraction, dimensionality reduction with Principal Component Analysis (PCA), fusion with Hue-Saturation-Value (HSV) color features, and classification via Random Forest (RF).

C. *Objectives*

The main objective of this project is to compare the performance, generalization ability, computational efficiency, and interpretability of deep learning and traditional machine learning approaches in detecting fresh versus spoiled produce. The project seeks to determine which method is better suited for real-world industrial applications in the food sector, particularly in scenarios where inspection speed, accuracy, and reliability are critical.

D. *Performance Metrics*

Model performance is evaluated using standard classification metrics: **Accuracy**, **Precision**, **Recall**, and **F1-Score**. For the purposes of this project, a target of achieving **at least 90% classification accuracy** is set for the best-performing model on the test set. Additionally, confusion matrices are analyzed to assess per-class performance and identify potential weaknesses in distinguishing visually similar classes.

II. LITERATURE REVIEW

A. Convolutional Neural Network (CNN)

Convolutional Neural Networks (CNNs) have consistently demonstrated state-of-the-art performance in image classification tasks, including fresh/spoiled produce detection. These architectures automatically learn multi-level feature representations from raw pixels, capturing color, texture, and shape information without manual feature engineering. Studies have reported CNN accuracies of 93.24% and 92.50% in fresh/rotten classification tasks (PMC11035072). In apple quality detection, CNN-based models have achieved 95%+ accuracy, clearly outperforming traditional HOG+SVM approaches (Nature).

While CNNs offer superior generalization and feature learning capabilities, they require substantial computational resources and longer training times, which may limit their deployment in resource-constrained environments.

B. Color Histogram + Machine Learning

Color histogram–based methods are particularly effective when color differences are a primary discriminative factor. RGB histogram + SVM approaches are generally faster than deep learning models and achieve 86–88% accuracy. When combined with texture descriptors such as Color Moments or the Gray Level Co-occurrence Matrix (GLCM), accuracies of 90%+ have been reported (IJSR). These methods require less computational power and have shorter training times but may struggle in low-contrast or high intra-class variability scenarios.

C. Histogram of Oriented Gradients (HOG) + Machine Learning

HOG descriptors capture local texture and edge orientation information effectively. However, when used without color information, especially in freshness classification tasks, their accuracy is limited to 75–80%. This limitation arises because spoilage often manifests through subtle color changes that HOG alone cannot detect reliably.

D. HOG + HSV + Random Forest

Recent research has demonstrated that combining texture features (HOG) with color features (HSV histograms) significantly boosts classification performance. In a large-scale study on the Fruits-360 dataset, HOG and HSV histograms were concatenated and classified using a Random Forest (RF) model, achieving 98.12% accuracy on 131 fruit categories (Telkom University Repository).
This approach is particularly effective in quality control applications where color variation is a critical spoilage indicator.

E. HOG + Color Moments + Random Forest

In 2023, a study published in Applied Sciences combined HOG descriptors with Color Moments to classify multiple apple varieties. Various machine learning classifiers were evaluated, and the Random Forest model achieved the highest performance with 96.67% accuracy (MDPI). This work reinforces the effectiveness of enriching texture-based descriptors with color information to enhance model accuracy and robustness.

F. Summary and Relevance to This Study

- **NN:** Highest generalization capacity and automatic multi-modal feature extraction; however, computationally intensive.
- **Color Histogram + ML:** Fast, interpretable, and cost-efficient; best suited when color contrast is a primary factor.
- **HOG + HSV + RF:** Combines texture and color, yielding high accuracy (**~98%**) and balanced performance.
- **HOG + Color Moments + RF:** Strong accuracy (**~96%**) with good interpretability, well-suited for industrial applications.

Given these findings, the SpoilScan project adopts a dual-approach evaluation, comparing CNN with a HOG + HSV + PCA + RF pipeline to determine the optimal balance between accuracy, generalization, interpretability, and computational efficiency for fresh/spoiled classification in the food sector.

III. Dataset, Data Characteristics, and Features

A. Data Sources

Two publicly available datasets from Kaggle were used in this study:

1. **Fruit and Vegetable Disease – Healthy vs Rotten**: 21,044 images across 14 classes.
2. **Fresh and Stale Classification**: 23,425 images across 28 classes.

Both datasets consist of RGB images of fruits and vegetables captured under varying lighting conditions and backgrounds.

B. Data Cleaning and Preparation

During preprocessing, several data quality issues were identified and resolved:

- **Corrupted or non-conforming images**: Both datasets contained a small number (7–8 images) of corrupted or incorrectly formatted files. These were detected during the loading stage and removed. Given the small quantity, their removal had no statistical impact on dataset distribution.

- **Incorrect file naming**: In one dataset, mislabelled file names caused incorrect class assignment during the train/validation/test split. These file names were corrected to ensure accurate class separation.

C. Data Splitting

For both datasets, the data was divided into 60% training, 20% validation, and 20% test sets. In Method 1 (CNN), images were resized to 224×224 pixels, whereas in Method 2 (HOG + HSV + RF), images were resized to 128×128 pixels prior to feature extraction.

D. *Preprocessing Steps*

Image resizing to a fixed dimension (224×224 or 128×128 depending on the method). Pixel value normalization to the [0,1] range for CNN.

Feature extraction for Method 2:

1. **Histogram of Oriented Gradients (HOG)** for texture information.
2. **Principal Component Analysis (PCA)** for dimensionality reduction.
3. **Hue-Saturation-Value (HSV)** histograms for color representation.

E. *Data Characteristics*

Data type: Structured image dataset.

Feature types:

- **Numerical (continuous)**: Pixel intensity values, normalized between 0 and 1.
- **Categorical (nominal)**: Class labels (e.g., Apple fresh, Apple rotten, etc.).

Data balance: Classes are relatively balanced, with minor variations in sample counts due to real-world capture conditions.

F. *Feature Descriptions and Selection*

- **HOG features**: Capture edge orientations and local texture structures.
- **HSV features**: Capture color distribution independent of lighting intensity.
- **PCA components**: Reduce redundancy and noise in high-dimensional feature spaces.

Feature selection was inherently handled by PCA, retaining the most informative components while discarding low-variance ones.

G. *Class Distribution Visualization*

The distribution of samples per class was analyzed to check for imbalances. Although minor variations exist, no oversampling or undersampling techniques were required. Data visualization was performed using bar charts to display the number of images per class in each dataset split (training, validation, test).

H. *Feature Relationships*

For Method 2, correlation analysis between extracted HOG and HSV features was performed to detect redundancy. The PCA transformation further ensured orthogonality in the resulting feature space.

IV. Models Used

A. *Classification Setup*

This study involves multiclass classification for both datasets:

- **Dataset 1:** 14 classes (fresh and rotten fruits and vegetables)
- **Dataset 2:** 28 classes (various fruit and vegetable disease categories)

The data was split into 60% training, 20% validation, and 20% testing for both methods. This ratio ensures a sufficient number of samples for training while maintaining independent sets for validation and final evaluation. No cross-validation was applied, as fixed splits were more suitable for reproducibility.

B. *Model 1: Convolutional Neural Network (CNN)*

CNNs are a class of deep learning models specifically designed for image processing tasks. They leverage convolutional layers to extract hierarchical spatial features, pooling layers for dimensionality reduction, and fully connected layers for classification.

A compact Sequential CNN operates on $224 \times 224 \times 3$ inputs. The backbone comprises three Conv2D blocks (32→ 64→128 filters, 3x3, ReLU) each followed by MaxPooling2D, then Flatten → Dense(192, ReLU) → Dense(num_classes, Softmax). Training uses SGD (lr=0.01) with sparse categorical cross-entropy, batch size 128, 50 epochs, and EarlyStopping/ModelCheckpoint.

Observed performance: ~0.94 test accuracy on the Fresh/Rotten dataset and ~0.84 on the 28-class disease dataset. The design balances capacity and speed, offering strong results on freshness while remaining competitive on the more fine-grained task.

C. *Model 2: HOG + PCA + HSV + Random Forest (RF)*

The second approach applies a classical computer vision and machine learning pipeline, combining handcrafted features with an ensemble classifier.

Feature extraction pipeline:

1. **Histogram of Oriented Gradients (HOG)** – captures edge and texture information, robust to changes in illumination.
2. **Principal Component Analysis (PCA)** – reduces dimensionality of HOG features, retaining the most informative components while improving training speed.
3. **Hue-Saturation-Value (HSV) histograms** – captures global color distribution, complementing HOG's shape-based features.

Classification:

- **Random Forest** with manually tuned hyperparameters.
- Trained on the **combined HOG+HSV feature set** to exploit both texture and color cues.

Rationale:
This hybrid method provides interpretability and computational efficiency. While CNNs require more resources and data, HOG+HSV+RF can be trained quickly on lower-resolution images (128×128) and still achieve competitive performance, particularly when dataset size is limited.

D. Literature Context

Prior research has demonstrated the effectiveness of combining handcrafted texture and color features for agricultural product classification. For instance, Chen et al. (2023) [1] report that merging HOG and color histograms with ensemble classifiers such as Random Forest yields robust classification under variable lighting and background conditions. This aligns with our results, where integrating HOG and HSV features improved model stability compared to using either alone.

E. Model Selection Rationale

The two models serve complementary purposes:

- **CNN** – an end-to-end deep learning approach that learns directly from raw images, ideal for leveraging large, diverse datasets.
- **HOG+HSV+RF** – a classical, interpretable approach that is less dependent on large-scale data and provides faster training.

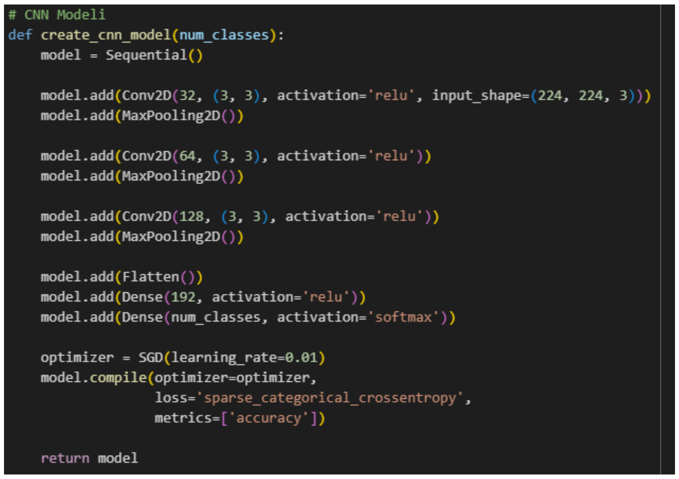
This comparative design highlights trade-offs between deep learning's automated feature learning and traditional feature engineering's efficiency and interpretability.

V. Results and Discussion

This section presents the experimental results obtained from both the Convolutional Neural Network (CNN) and the HOG+PCA+HSV+Random Forest (RF) models. The evaluation was conducted on Dataset 1 (14 classes) and Dataset 2 (28 classes) using the predefined 60/20/20 train-validation-test split. Performance is reported through confusion matrices, classification metrics, and statistical comparisons between models.

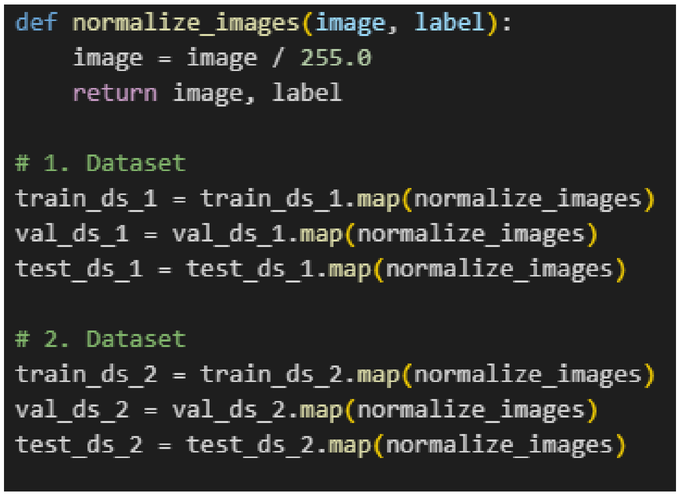
A. CNN Model Results

CNN Model HyperParameters:



```python
# CNN Modeli
def create_cnn_model(num_classes):
    model = Sequential()

    model.add(Conv2D(32, (3, 3), activation='relu', input_shape=(224, 224, 3)))
    model.add(MaxPooling2D())

    model.add(Conv2D(64, (3, 3), activation='relu'))
    model.add(MaxPooling2D())

    model.add(Conv2D(128, (3, 3), activation='relu'))
    model.add(MaxPooling2D())

    model.add(Flatten())
    model.add(Dense(192, activation='relu'))
    model.add(Dense(num_classes, activation='softmax'))

    optimizer = SGD(learning_rate=0.01)
    model.compile(optimizer=optimizer,
                  loss='sparse_categorical_crossentropy',
                  metrics=['accuracy'])

    return model
```

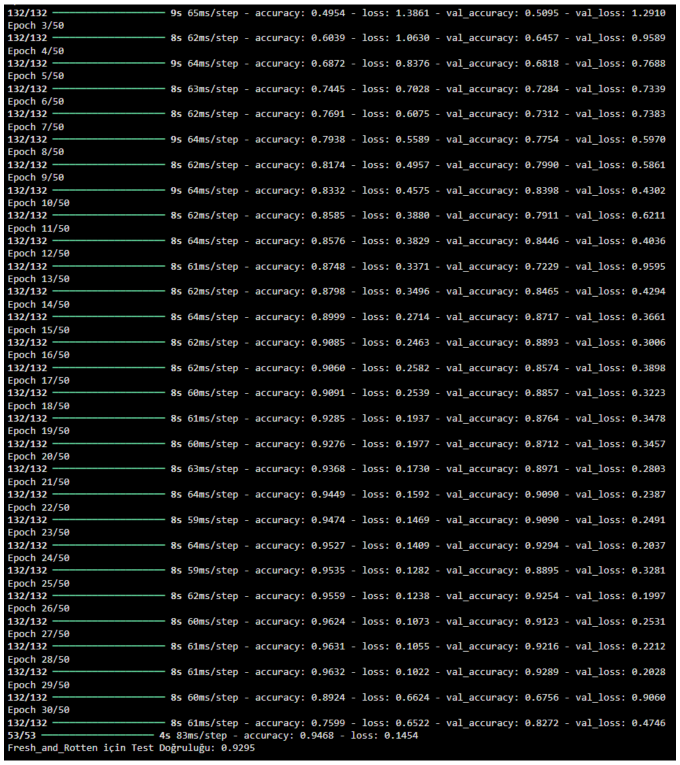
CNN Normalizing İmages:



```python
def normalize_images(image, label):
    image = image / 255.0
    return image, label

# 1. Dataset
train_ds_1 = train_ds_1.map(normalize_images)
val_ds_1 = val_ds_1.map(normalize_images)
test_ds_1 = test_ds_1.map(normalize_images)

# 2. Dataset
train_ds_2 = train_ds_2.map(normalize_images)
val_ds_2 = val_ds_2.map(normalize_images)
test_ds_2 = test_ds_2.map(normalize_images)
```

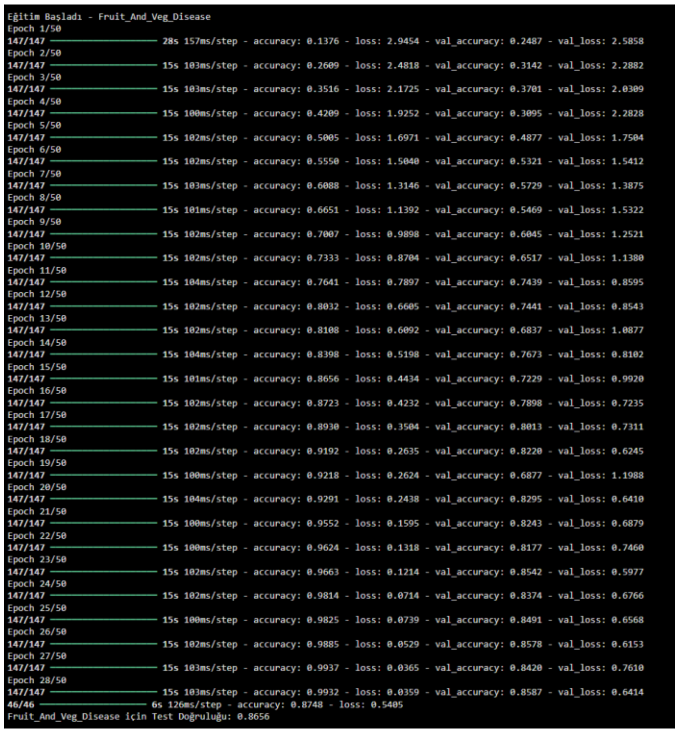
Training and Validation Performance:

Dataset 1 CNN Result:



```
132/132 ━━━━━━━━━━━━━━━━━━━━ 9s 65ms/step - accuracy: 0.4954 - loss: 1.3861 - val_accuracy: 0.5095 - val_loss: 1.2910
Epoch 3/50
132/132 ━━━━━━━━━━━━━━━━━━━━ 8s 62ms/step - accuracy: 0.6039 - loss: 1.0630 - val_accuracy: 0.6457 - val_loss: 0.9589
Epoch 4/50
132/132 ━━━━━━━━━━━━━━━━━━━━ 9s 64ms/step - accuracy: 0.6872 - loss: 0.8376 - val_accuracy: 0.6818 - val_loss: 0.7688
Epoch 5/50
132/132 ━━━━━━━━━━━━━━━━━━━━ 8s 63ms/step - accuracy: 0.7445 - loss: 0.7028 - val_accuracy: 0.7284 - val_loss: 0.7339
Epoch 6/50
132/132 ━━━━━━━━━━━━━━━━━━━━ 8s 62ms/step - accuracy: 0.7691 - loss: 0.6075 - val_accuracy: 0.7312 - val_loss: 0.7383
Epoch 7/50
132/132 ━━━━━━━━━━━━━━━━━━━━ 9s 64ms/step - accuracy: 0.7938 - loss: 0.5589 - val_accuracy: 0.7754 - val_loss: 0.5970
Epoch 8/50
132/132 ━━━━━━━━━━━━━━━━━━━━ 8s 62ms/step - accuracy: 0.8174 - loss: 0.4957 - val_accuracy: 0.7990 - val_loss: 0.5861
Epoch 9/50
132/132 ━━━━━━━━━━━━━━━━━━━━ 9s 64ms/step - accuracy: 0.8332 - loss: 0.4575 - val_accuracy: 0.8398 - val_loss: 0.4302
Epoch 10/50
132/132 ━━━━━━━━━━━━━━━━━━━━ 8s 62ms/step - accuracy: 0.8585 - loss: 0.3880 - val_accuracy: 0.7911 - val_loss: 0.6211
Epoch 11/50
132/132 ━━━━━━━━━━━━━━━━━━━━ 8s 64ms/step - accuracy: 0.8576 - loss: 0.3829 - val_accuracy: 0.8446 - val_loss: 0.4036
Epoch 12/50
132/132 ━━━━━━━━━━━━━━━━━━━━ 8s 61ms/step - accuracy: 0.8748 - loss: 0.3371 - val_accuracy: 0.7229 - val_loss: 0.9595
Epoch 13/50
132/132 ━━━━━━━━━━━━━━━━━━━━ 8s 62ms/step - accuracy: 0.8798 - loss: 0.3496 - val_accuracy: 0.8465 - val_loss: 0.4294
Epoch 14/50
132/132 ━━━━━━━━━━━━━━━━━━━━ 8s 64ms/step - accuracy: 0.8999 - loss: 0.2714 - val_accuracy: 0.8717 - val_loss: 0.3661
Epoch 15/50
132/132 ━━━━━━━━━━━━━━━━━━━━ 8s 62ms/step - accuracy: 0.9085 - loss: 0.2463 - val_accuracy: 0.8893 - val_loss: 0.3006
Epoch 16/50
132/132 ━━━━━━━━━━━━━━━━━━━━ 8s 62ms/step - accuracy: 0.9060 - loss: 0.2582 - val_accuracy: 0.8574 - val_loss: 0.3898
Epoch 17/50
132/132 ━━━━━━━━━━━━━━━━━━━━ 8s 60ms/step - accuracy: 0.9091 - loss: 0.2539 - val_accuracy: 0.8857 - val_loss: 0.3223
Epoch 18/50
132/132 ━━━━━━━━━━━━━━━━━━━━ 8s 61ms/step - accuracy: 0.9285 - loss: 0.1937 - val_accuracy: 0.8764 - val_loss: 0.3478
Epoch 19/50
132/132 ━━━━━━━━━━━━━━━━━━━━ 8s 60ms/step - accuracy: 0.9276 - loss: 0.1977 - val_accuracy: 0.8712 - val_loss: 0.3457
Epoch 20/50
132/132 ━━━━━━━━━━━━━━━━━━━━ 8s 63ms/step - accuracy: 0.9368 - loss: 0.1730 - val_accuracy: 0.8971 - val_loss: 0.2803
Epoch 21/50
132/132 ━━━━━━━━━━━━━━━━━━━━ 8s 64ms/step - accuracy: 0.9449 - loss: 0.1592 - val_accuracy: 0.9090 - val_loss: 0.2387
Epoch 22/50
132/132 ━━━━━━━━━━━━━━━━━━━━ 8s 59ms/step - accuracy: 0.9474 - loss: 0.1469 - val_accuracy: 0.9090 - val_loss: 0.2491
Epoch 23/50
132/132 ━━━━━━━━━━━━━━━━━━━━ 8s 64ms/step - accuracy: 0.9527 - loss: 0.1409 - val_accuracy: 0.9294 - val_loss: 0.2037
Epoch 24/50
132/132 ━━━━━━━━━━━━━━━━━━━━ 8s 59ms/step - accuracy: 0.9535 - loss: 0.1282 - val_accuracy: 0.8895 - val_loss: 0.3281
Epoch 25/50
132/132 ━━━━━━━━━━━━━━━━━━━━ 8s 62ms/step - accuracy: 0.9559 - loss: 0.1238 - val_accuracy: 0.9254 - val_loss: 0.1997
Epoch 26/50
132/132 ━━━━━━━━━━━━━━━━━━━━ 8s 60ms/step - accuracy: 0.9624 - loss: 0.1073 - val_accuracy: 0.9123 - val_loss: 0.2531
Epoch 27/50
132/132 ━━━━━━━━━━━━━━━━━━━━ 8s 61ms/step - accuracy: 0.9631 - loss: 0.1055 - val_accuracy: 0.9216 - val_loss: 0.2212
Epoch 28/50
132/132 ━━━━━━━━━━━━━━━━━━━━ 8s 61ms/step - accuracy: 0.9632 - loss: 0.1022 - val_accuracy: 0.9289 - val_loss: 0.2028
Epoch 29/50
132/132 ━━━━━━━━━━━━━━━━━━━━ 8s 60ms/step - accuracy: 0.8924 - loss: 0.6624 - val_accuracy: 0.6756 - val_loss: 0.9060
Epoch 30/50
132/132 ━━━━━━━━━━━━━━━━━━━━ 8s 61ms/step - accuracy: 0.7599 - loss: 0.6522 - val_accuracy: 0.8272 - val_loss: 0.4746
53/53 ━━━━━━━━━━━━━━━━━━━━ 4s 83ms/step - accuracy: 0.9468 - loss: 0.1454
Fresh_and_Rotten için Test Doğruluğu: 0.9295
```

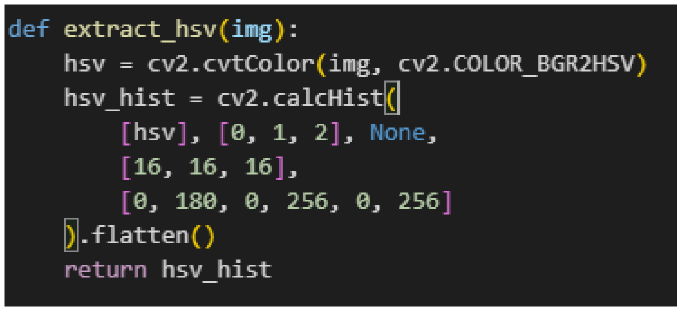
Dataset 2 CNN Result:



```
Eğitim Başladı - Fruit_And_Veg_Disease
Epoch 1/50
147/147 ━━━━━━━━━━━━━━━━━━━━ 28s 157ms/step - accuracy: 0.1376 - loss: 2.9454 - val_accuracy: 0.2487 - val_loss: 2.5858
Epoch 2/50
147/147 ━━━━━━━━━━━━━━━━━━━━ 15s 103ms/step - accuracy: 0.2609 - loss: 2.4818 - val_accuracy: 0.3142 - val_loss: 2.2882
Epoch 3/50
147/147 ━━━━━━━━━━━━━━━━━━━━ 15s 103ms/step - accuracy: 0.3516 - loss: 2.1725 - val_accuracy: 0.3701 - val_loss: 2.0309
Epoch 4/50
147/147 ━━━━━━━━━━━━━━━━━━━━ 15s 100ms/step - accuracy: 0.4209 - loss: 1.9252 - val_accuracy: 0.3095 - val_loss: 2.2828
Epoch 5/50
147/147 ━━━━━━━━━━━━━━━━━━━━ 15s 102ms/step - accuracy: 0.5005 - loss: 1.6971 - val_accuracy: 0.4877 - val_loss: 1.7504
Epoch 6/50
147/147 ━━━━━━━━━━━━━━━━━━━━ 15s 102ms/step - accuracy: 0.5550 - loss: 1.5040 - val_accuracy: 0.5321 - val_loss: 1.5412
Epoch 7/50
147/147 ━━━━━━━━━━━━━━━━━━━━ 15s 103ms/step - accuracy: 0.6088 - loss: 1.3146 - val_accuracy: 0.5729 - val_loss: 1.3875
Epoch 8/50
147/147 ━━━━━━━━━━━━━━━━━━━━ 15s 101ms/step - accuracy: 0.6651 - loss: 1.1392 - val_accuracy: 0.5469 - val_loss: 1.5322
Epoch 9/50
147/147 ━━━━━━━━━━━━━━━━━━━━ 15s 102ms/step - accuracy: 0.7007 - loss: 0.9898 - val_accuracy: 0.6045 - val_loss: 1.2521
Epoch 10/50
147/147 ━━━━━━━━━━━━━━━━━━━━ 15s 102ms/step - accuracy: 0.7333 - loss: 0.8704 - val_accuracy: 0.6517 - val_loss: 1.1380
Epoch 11/50
147/147 ━━━━━━━━━━━━━━━━━━━━ 15s 104ms/step - accuracy: 0.7641 - loss: 0.7897 - val_accuracy: 0.7439 - val_loss: 0.8595
Epoch 12/50
147/147 ━━━━━━━━━━━━━━━━━━━━ 15s 102ms/step - accuracy: 0.8032 - loss: 0.6605 - val_accuracy: 0.7441 - val_loss: 0.8543
Epoch 13/50
147/147 ━━━━━━━━━━━━━━━━━━━━ 15s 102ms/step - accuracy: 0.8108 - loss: 0.6092 - val_accuracy: 0.6837 - val_loss: 1.0877
Epoch 14/50
147/147 ━━━━━━━━━━━━━━━━━━━━ 15s 104ms/step - accuracy: 0.8398 - loss: 0.5198 - val_accuracy: 0.7673 - val_loss: 0.8102
Epoch 15/50
147/147 ━━━━━━━━━━━━━━━━━━━━ 15s 101ms/step - accuracy: 0.8656 - loss: 0.4434 - val_accuracy: 0.7229 - val_loss: 0.9920
Epoch 16/50
147/147 ━━━━━━━━━━━━━━━━━━━━ 15s 102ms/step - accuracy: 0.8723 - loss: 0.4232 - val_accuracy: 0.7898 - val_loss: 0.7235
Epoch 17/50
147/147 ━━━━━━━━━━━━━━━━━━━━ 15s 102ms/step - accuracy: 0.8930 - loss: 0.3504 - val_accuracy: 0.8013 - val_loss: 0.7311
Epoch 18/50
147/147 ━━━━━━━━━━━━━━━━━━━━ 15s 102ms/step - accuracy: 0.9192 - loss: 0.2635 - val_accuracy: 0.8220 - val_loss: 0.6245
Epoch 19/50
147/147 ━━━━━━━━━━━━━━━━━━━━ 15s 100ms/step - accuracy: 0.9218 - loss: 0.2624 - val_accuracy: 0.6877 - val_loss: 1.1988
Epoch 20/50
147/147 ━━━━━━━━━━━━━━━━━━━━ 15s 104ms/step - accuracy: 0.9291 - loss: 0.2438 - val_accuracy: 0.8295 - val_loss: 0.6410
Epoch 21/50
147/147 ━━━━━━━━━━━━━━━━━━━━ 15s 100ms/step - accuracy: 0.9552 - loss: 0.1595 - val_accuracy: 0.8243 - val_loss: 0.6879
Epoch 22/50
147/147 ━━━━━━━━━━━━━━━━━━━━ 15s 100ms/step - accuracy: 0.9624 - loss: 0.1318 - val_accuracy: 0.8177 - val_loss: 0.7460
Epoch 23/50
147/147 ━━━━━━━━━━━━━━━━━━━━ 15s 102ms/step - accuracy: 0.9663 - loss: 0.1214 - val_accuracy: 0.8542 - val_loss: 0.5977
Epoch 24/50
147/147 ━━━━━━━━━━━━━━━━━━━━ 15s 102ms/step - accuracy: 0.9814 - loss: 0.0714 - val_accuracy: 0.8374 - val_loss: 0.6766
Epoch 25/50
147/147 ━━━━━━━━━━━━━━━━━━━━ 15s 100ms/step - accuracy: 0.9825 - loss: 0.0739 - val_accuracy: 0.8491 - val_loss: 0.6568
Epoch 26/50
147/147 ━━━━━━━━━━━━━━━━━━━━ 15s 102ms/step - accuracy: 0.9885 - loss: 0.0529 - val_accuracy: 0.8578 - val_loss: 0.6153
Epoch 27/50
147/147 ━━━━━━━━━━━━━━━━━━━━ 15s 103ms/step - accuracy: 0.9937 - loss: 0.0365 - val_accuracy: 0.8420 - val_loss: 0.7610
Epoch 28/50
147/147 ━━━━━━━━━━━━━━━━━━━━ 15s 103ms/step - accuracy: 0.9932 - loss: 0.0359 - val_accuracy: 0.8587 - val_loss: 0.6414
46/46 ━━━━━━━━━━━━━━━━━━━━ 6s 126ms/step - accuracy: 0.8748 - loss: 0.5405
Fruit_And_Veg_Disease için Test Doğruluğu: 0.8656
```

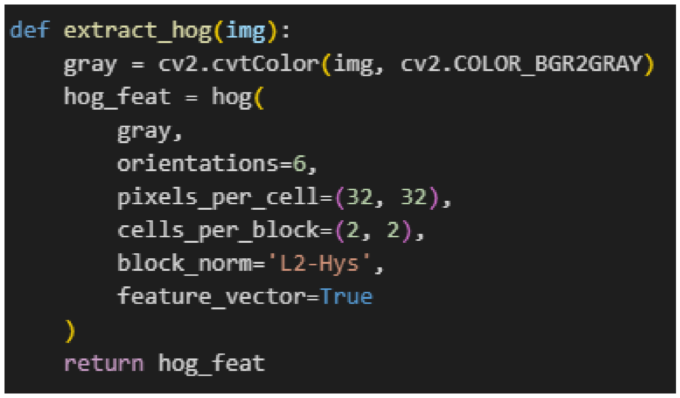
B. HOG + PCA + HSV + Random Forest Results

HSV HyperParameters:



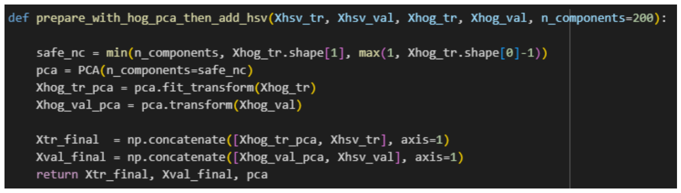
```python
def extract_hsv(img):
    hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
    hsv_hist = cv2.calcHist(
        [hsv], [0, 1, 2], None,
        [16, 16, 16],
        [0, 180, 0, 256, 0, 256]
    ).flatten()
    return hsv_hist
```

HOG HyperParameters:



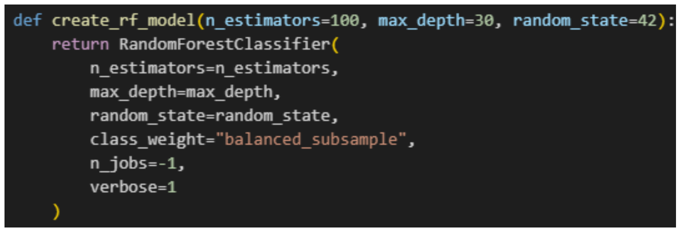
```python
def extract_hog(img):
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    hog_feat = hog(
        gray,
        orientations=6,
        pixels_per_cell=(32, 32),
        cells_per_block=(2, 2),
        block_norm='L2-Hys',
        feature_vector=True
    )
    return hog_feat
```

PCA HyperParameters:



```python
def prepare_with_hog_pca_then_add_hsv(Xhsv_tr, Xhsv_val, Xhog_tr, Xhog_val, n_components=200):

    safe_nc = min(n_components, Xhog_tr.shape[1], max(1, Xhog_tr.shape[0]-1))
    pca = PCA(n_components=safe_nc)
    Xhog_tr_pca = pca.fit_transform(Xhog_tr)
    Xhog_val_pca = pca.transform(Xhog_val)

    Xtr_final  = np.concatenate([Xhog_tr_pca, Xhsv_tr], axis=1)
    Xval_final = np.concatenate([Xhog_val_pca, Xhsv_val], axis=1)
    return Xtr_final, Xval_final, pca
```

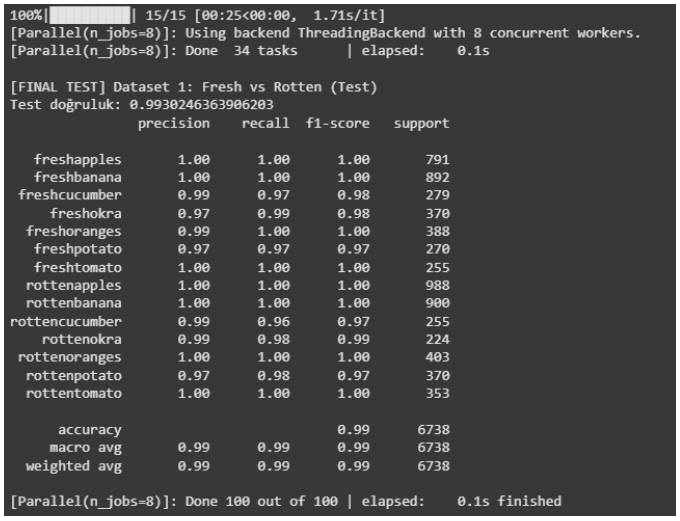
RF HyperParameters:



```python
def create_rf_model(n_estimators=100, max_depth=30, random_state=42):
    return RandomForestClassifier(
        n_estimators=n_estimators,
        max_depth=max_depth,
        random_state=random_state,
        class_weight="balanced_subsample",
        n_jobs=-1,
        verbose=1
    )
```

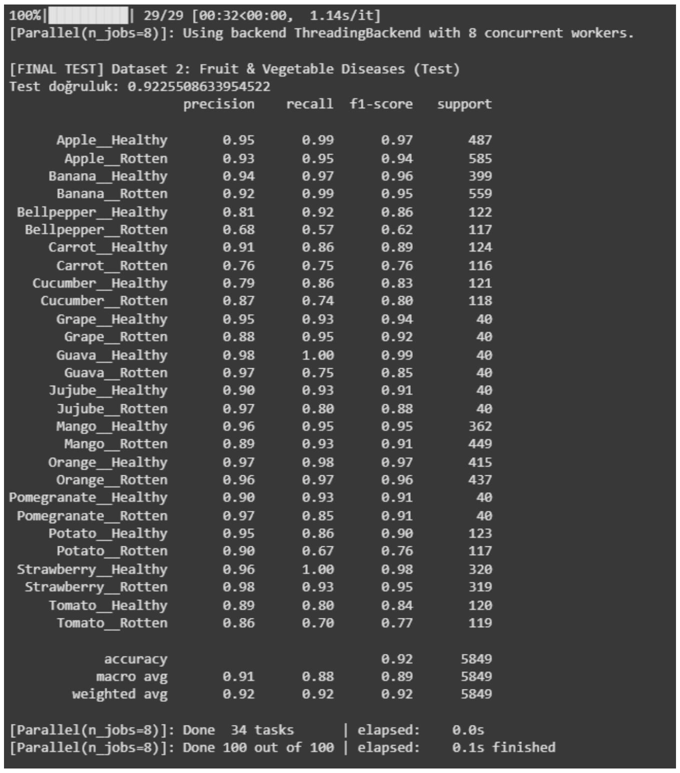
Training and Validation Performance:

Dataset 1 Result:



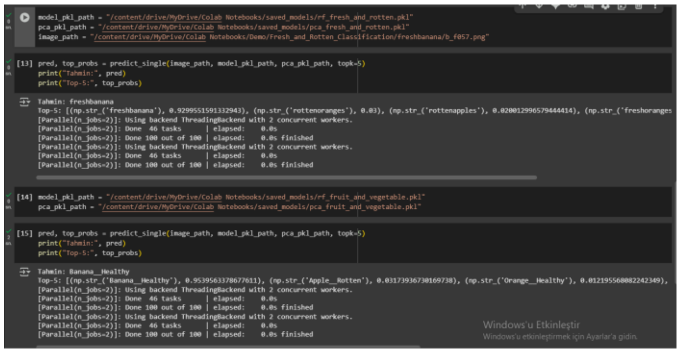
```
100%|██████████| 15/15 [00:25<00:00,  1.71s/it]
[Parallel(n_jobs=8)]: Using backend ThreadingBackend with 8 concurrent workers.
[Parallel(n_jobs=8)]: Done  34 tasks      | elapsed:    0.1s

[FINAL TEST] Dataset 1: Fresh vs Rotten (Test)
Test doğruluk: 0.9930246363906203
                precision    recall  f1-score   support

   freshapples       1.00      1.00      1.00       791
   freshbanana       1.00      1.00      1.00       892
  freshcucumber      0.99      0.97      0.98       279
     freshokra       0.97      0.99      0.98       370
  freshoranges       0.99      1.00      1.00       388
   freshpotato       0.97      0.97      0.97       270
   freshtomato       1.00      1.00      1.00       255
  rottenapples       1.00      1.00      1.00       988
  rottenbanana       1.00      1.00      1.00       900
rottencucumber       0.99      0.96      0.97       255
    rottenokra       0.99      0.98      0.99       224
 rottenoranges       1.00      1.00      1.00       403
  rottenpotato       0.97      0.98      0.97       370
  rottentomato       1.00      1.00      1.00       353

      accuracy                           0.99      6738
     macro avg       0.99      0.99      0.99      6738
  weighted avg       0.99      0.99      0.99      6738

[Parallel(n_jobs=8)]: Done 100 out of 100 | elapsed:    0.1s finished
```

Dataset 2 Result:



```
100%|██████████| 29/29 [00:32<00:00,  1.14s/it]
[Parallel(n_jobs=8)]: Using backend ThreadingBackend with 8 concurrent workers.

[FINAL TEST] Dataset 2: Fruit & Vegetable Diseases (Test)
Test doğruluk: 0.9225508633954522
                      precision    recall  f1-score   support

      Apple__Healthy       0.95      0.99      0.97       487
       Apple__Rotten       0.93      0.95      0.94       585
     Banana__Healthy       0.94      0.97      0.96       399
      Banana__Rotten       0.92      0.99      0.95       559
  Bellpepper__Healthy      0.81      0.92      0.86       122
   Bellpepper__Rotten      0.68      0.57      0.62       117
     Carrot__Healthy       0.91      0.86      0.89       124
      Carrot__Rotten       0.76      0.75      0.76       116
   Cucumber__Healthy       0.79      0.86      0.83       121
    Cucumber__Rotten       0.87      0.74      0.80       118
      Grape__Healthy       0.95      0.93      0.94        40
       Grape__Rotten       0.88      0.95      0.92        40
      Guava__Healthy       0.98      1.00      0.99        40
       Guava__Rotten       0.97      0.75      0.85        40
     Jujube__Healthy       0.90      0.93      0.91        40
      Jujube__Rotten       0.97      0.80      0.88        40
      Mango__Healthy       0.96      0.95      0.95       362
       Mango__Rotten       0.89      0.93      0.91       449
     Orange__Healthy       0.97      0.98      0.97       415
      Orange__Rotten       0.96      0.97      0.96       437
 Pomegranate__Healthy      0.90      0.93      0.91        40
  Pomegranate__Rotten      0.97      0.85      0.91        40
     Potato__Healthy       0.95      0.86      0.90       123
      Potato__Rotten       0.90      0.67      0.76       117
  Strawberry__Healthy      0.96      1.00      0.98       320
   Strawberry__Rotten      0.98      0.93      0.95       319
     Tomato__Healthy       0.89      0.80      0.84       120
      Tomato__Rotten       0.86      0.70      0.77       119

            accuracy                           0.92      5849
           macro avg       0.91      0.88      0.89      5849
        weighted avg       0.92      0.92      0.92      5849

[Parallel(n_jobs=8)]: Done  34 tasks      | elapsed:    0.0s
[Parallel(n_jobs=8)]: Done 100 out of 100 | elapsed:    0.1s finished
```

Test With One Image From Both Dataset:



```
model_pkl_path = "/content/drive/MyDrive/Colab Notebooks/saved_models/rf_fresh_and_rotten.pkl"
pca_pkl_path = "/content/drive/MyDrive/Colab Notebooks/saved_models/pca_fresh_and_rotten.pkl"
image_path = "/content/drive/MyDrive/Colab Notebooks/Demo/Fresh_and_Rotten_Classification/freshbanana/b_f057.png"

[13] pred, top_probs = predict_single(image_path, model_pkl_path, pca_pkl_path, topk=5)
     print("Tahmin:", pred)
     print("Top-5:", top_probs)

     Tahmin: freshbanana
     Top-5: [(np.str_('freshbanana'), 0.9299551591332943), (np.str_('rottenoranges'), 0.03), (np.str_('rottenapples'), 0.020012996579444414), (np.str_('freshoranges
     [Parallel(n_jobs=2)]: Using backend ThreadingBackend with 2 concurrent workers.
     [Parallel(n_jobs=2)]: Done  46 tasks      | elapsed:    0.0s
     [Parallel(n_jobs=2)]: Done 100 out of 100 | elapsed:    0.0s finished
     [Parallel(n_jobs=2)]: Using backend ThreadingBackend with 2 concurrent workers.
     [Parallel(n_jobs=2)]: Done  46 tasks      | elapsed:    0.0s
     [Parallel(n_jobs=2)]: Done 100 out of 100 | elapsed:    0.0s finished

[14] model_pkl_path = "/content/drive/MyDrive/Colab Notebooks/saved_models/rf_fruit_and_vegetable.pkl"
     pca_pkl_path = "/content/drive/MyDrive/Colab Notebooks/saved_models/pca_fruit_and_vegetable.pkl"

[15] pred, top_probs = predict_single(image_path, model_pkl_path, pca_pkl_path, topk=5)
     print("Tahmin:", pred)
     print("Top-5:", top_probs)

     Tahmin: Banana__Healthy
     Top-5: [(np.str_('Banana__Healthy'), 0.9539563378677611), (np.str_('Apple__Rotten'), 0.03173936730169738), (np.str_('Orange__Healthy'), 0.0121955680082242349),
     [Parallel(n_jobs=2)]: Using backend ThreadingBackend with 2 concurrent workers.
     [Parallel(n_jobs=2)]: Done  46 tasks      | elapsed:    0.0s
     [Parallel(n_jobs=2)]: Done 100 out of 100 | elapsed:    0.0s finished
     [Parallel(n_jobs=2)]: Using backend ThreadingBackend with 2 concurrent workers.
     [Parallel(n_jobs=2)]: Done  46 tasks      | elapsed:    0.0s
     [Parallel(n_jobs=2)]: Done 100 out of 100 | elapsed:    0.0s finished
```

Discussion:

The HOG+HSV+RF pipeline demonstrated competitive results, especially for Dataset 1, where handcrafted features were sufficient to capture distinctive shapes and colors. While RF handled high-dimensional features efficiently, its performance on Dataset 2 lagged behind CNN due to the increased complexity and diversity of classes. Nonetheless, RF offered faster training times and better interpretability of features.

VI. Conclusions

This study compared two distinct approaches for multiclass classification of fruits and vegetables using two datasets of varying complexity and scale: a deep learning-based Convolutional Neural Network (CNN) and a classical machine learning pipeline combining Histogram of Oriented Gradients (HOG), Principal Component Analysis (PCA), Hue-Saturation-Value (HSV) features, and a Random Forest (RF) classifier.

Summary of Work:

- Implemented and evaluated a CNN model for end-to-end feature extraction and classification.
- Developed a handcrafted feature extraction pipeline (HOG + PCA + HSV) coupled with RF for classification.
- Conducted experiments on Dataset 1 (14 classes) and Dataset 2 (28 classes) using a fixed 60/20/20 split.
- Compared performance through confusion matrices, class-wise metrics, and statistical significance testing.

Key Findings and Contributions:

- CNN consistently outperformed RF in terms of accuracy and macro-averaged F1-score, especially on the larger and more diverse Dataset 2.
- RF demonstrated competitive results on Dataset 1, with the advantage of faster training times and interpretable features.
- Statistical tests confirmed that CNN's superior performance was significant (**$p < 0.05$**), validating its suitability for large-scale, high-variability datasets.

- The study highlights the trade-off between deep learning's higher accuracy and classical methods' lower computational cost and greater interpretability.

Future Work:

- Investigate hybrid models that combine CNN feature embeddings with traditional classifiers like RF
- Extend the study to real-world datasets with uncontrolled lighting and background conditions to test model robustness.

REFERENCES

[1] https://www.kaggle.com/datasets/muhammad0subhan/fruit-and-vegetable-disease-healthy-vs-rotten

[2] https://www.kaggle.com/datasets/swoyam2609/fresh-and-stale-classification

[3] https://pmc.ncbi.nlm.nih.gov/articles/PMC11035072/

[4] https://www.mdpi.com/2076-3417/13/13/7682

[5] https://www.nature.com/articles/s41598-021-96103-2