



# Comparative Analysis of Machine Learning Algorithms for Fruit Classification

**Yousif Salah Mohammed**

ID: 22P0232

**Sandra Ghayes Zarif**

ID: 22P0300

**Github:** [github.com/ysif9/fruit-detection](https://github.com/ysif9/fruit-detection)

**Dataset:** [huggingface.co/datasets/ysif9/fruit-recognition](https://huggingface.co/datasets/ysif9/fruit-recognition)

**Supervisor:** Dr. Alaa Hamdy

*Professor, Computer and Systems Engineering*

**Co-supervisor:** Eng. George Emad

*Teaching Assistant*

Eng. Engy Ahmed

*Teaching Assistant*

Ain Shams University

Faculty of Engineering

Computer Engineering & Software Systems

CSE381: Introduction to Machine Learning

*(Machine Learning Project 7)*

15 January 2026

# CONTENTS

<b>Contents</b>	<b>i</b>
<b>List of Figures</b>	<b>1</b>
<b>List of Tables</b>	<b>1</b>
<b>1 Introduction</b>	<b>2</b>
1.1 Problem Statement . . . . .	2
1.2 Project Objective . . . . .	2
1.3 Scope . . . . .	3
1.3.1 Dataset and Classes . . . . .	3
1.3.2 Modeling Approaches . . . . .	3
<b>2 Dataset and Preprocessing</b>	<b>4</b>
2.1 Data Source . . . . .	4
2.2 Class Distribution . . . . .	4
2.3 Dataset Splitting . . . . .	5
2.4 Preprocessing Pipeline . . . . .	5
2.4.1 Image Resizing . . . . .	5
2.4.2 Normalization . . . . .	6
<b>3 Feature Extraction &amp; Selection</b>	<b>7</b>
3.1 Base Feature Extraction (MobileNetV2) . . . . .	7
3.2 Dimensionality Reduction . . . . .	7
3.2.1 Principal Component Analysis (PCA) . . . . .	7
3.2.2 Fisher’s Linear Discriminant Analysis (LDA) . . . . .	8
<b>4 Classification Models</b>	<b>9</b>
4.1 Performance Metrics . . . . .	9
4.2 Decision Tree Classifier . . . . .	10
4.2.1 Performance Evaluation . . . . .	10
4.2.2 Confusion Matrix Analysis . . . . .	10
4.3 Random Forest Classifier . . . . .	11
4.3.1 Implementation Details . . . . .	11

4.3.2	Performance Evaluation . . . . .	11
4.3.3	Confusion Matrix Analysis . . . . .	11
4.4	XGBoost Classifier . . . . .	12
4.4.1	Implementation Details . . . . .	13
4.4.2	Performance Evaluation . . . . .	13
4.4.3	Confusion Matrix Analysis . . . . .	13
4.5	K-Nearest Neighbors (KNN) . . . . .	14
4.6	Support Vector Machine (SVM) . . . . .	15
4.7	Artificial Neural Networks (Deep Learning Approach) . . . . .	17
<b>5</b>	<b>Discussion &amp; Analysis</b>	<b>18</b>
<b>6</b>	<b>Conclusion &amp; Future Work</b>	<b>19</b>
6.1	Summary of Findings . . . . .	19
6.2	Future Improvements . . . . .	19

List of Figures

4.1

Confusion Matrix for Decision Tree Classifier.

10

4.2

Decision Tree Error Samples

11

4.3

Confusion Matrix for Random Forest Classifier.

12

4.4

Random Forest Error Samples

12

4.5

Confusion Matrix for XGBoost Classifier.

13

4.6

XGBoost Error Samples

14

4.7

KNN Classification Report

15

4.8

Comparison of SVM Classifiers

16

4.9

Custom Layer

17

4.10

Training & Validation Loss and Accuracy Curves

17

5.1

Dataset Sample

18

List of Tables

2.1

Dataset Partitioning

5

# INTRODUCTION

## 1.1 Problem Statement

The demand for automated visual recognition systems in the agricultural and retail sectors has grown significantly in recent years. Manual sorting and classification of fruits are labor-intensive, time-consuming, and prone to human error. In agricultural supply chains, efficient grading and sorting are crucial for quality control and logistics. Similarly, in the retail sector, the evolution of automated supermarkets and self-checkout stations requires intelligent systems capable of instantly identifying produce without barcodes.

The core problem addressed in this project is the development of a computer vision system capable of accurately classifying fruit images into specific categories. By automating this process, we address the need for scalability and consistency in food processing and point-of-sale systems, ultimately aiming to reduce operational costs and improve efficiency in fruit handling.

## 1.2 Project Objective

The primary objective of this project is to design, implement, and evaluate a robust fruit recognition system. Beyond achieving high classification accuracy, this project aims to provide a comprehensive comparative analysis between two distinct methodological paradigms in computer vision:

1. **Classical Machine Learning (Feature-based):** Utilizing a pre-trained feature extractor (MobileNetV2) to convert images into high-dimensional feature vectors, followed by the application of traditional classifiers such as Support Vector Machines (SVM), K-Nearest Neighbors (KNN), and ensemble methods.
2. **Deep Learning (End-to-End):** Implementing Convolutional Neural Networks (CNNs) to learn hierarchical feature representations directly from raw pixel data, exploring both custom architectures and transfer learning techniques.

The study seeks to evaluate these models not just on accuracy, but also on their ability to generalize across different fruit classes and their computational implications.

## 1.3 Scope

The scope of this project is defined by the following parameters, datasets, and modeling techniques:

### 1.3.1 Dataset and Classes

The system is trained and tested on the `ysif9/fruit-recognition` dataset (sourced from Hugging Face/Kaggle). The classification task is limited to **10 distinct fruit classes**, ensuring a diverse representation of colors, shapes, and textures:

- Apple, Banana, Orange, Mango, Grapes
- Pineapple, Watermelon, Pomegranate, Strawberry, Lemon

### 1.3.2 Modeling Approaches

To ensure a thorough evaluation, the project implements and compares **7 different classification models**:

- **Feature Extraction Approaches:**
  - **SVM (Support Vector Machine):** Investigating Linear SVM, PCA-reduced RBF SVM, and Fisher's Linear Discriminant Analysis (LDA) combined with SVM.
  - **KNN (K-Nearest Neighbors):** Utilizing distance-based classification in the feature space.
  - **Decision Tree:** Evaluating a single tree structure for baseline interpretability.
  - **Random Forest:** Applying bagging ensemble methods to improve generalization.
  - **XGBoost:** Utilizing gradient boosting for optimized performance on feature vectors.
- **Deep Learning Approaches:**
  - **Simple CNN:** A custom, lightweight Convolutional Neural Network trained from scratch.
  - **Transfer Learning:** A MobileNetV2 architecture fine-tuned for this specific domain.

## DATASET AND PREPROCESSING

The quality and preparation of data are fundamental to the success of any machine learning model. This chapter details the provenance of the data used in this project, the specific fruit classes selected, and the rigorous preprocessing pipelines established to transform raw images into trainable features. We employed distinct strategies for classical machine learning algorithms versus deep learning architectures to maximize the performance of each approach.

### 2.1 Data Source

The primary dataset for this project was hosted by us at the Hugging Face hub repository `ysif9/fruit-recognition`. This repository serves as a filtered and curated subset of the larger *"Fruit and Vegetables Classification"* dataset, originally hosted on Kaggle by Youssef Salah Zakria.

While the original Kaggle dataset provides a comprehensive collection of diverse food items, including various vegetables, this project required a focused scope on culinary fruits to test specific classification boundaries. Consequently, the data was filtered to isolate distinct fruit classes. The dataset contains images captured under various lighting conditions, angles, and backgrounds, providing a realistic challenge for image recognition systems.

### 2.2 Class Distribution

The dataset consists of 10 distinct classes of fruits. These classes were selected to represent a variety of shapes, colors, and textures, ranging from the distinct curves of bananas to the complex textures of pineapples and strawberries. The specific classes are:

- **Apple:** Characterized by round shape and red/green variations.
- **Banana:** distinct elongated shape and yellow color.
- **Orange:** Round, textured citrus skin.

- **Mango:** Oval shape with yellow/orange gradients.
- **Grapes:** Clustered small spheres.
- **Pineapple:** Complex, rough texture and distinct crown.
- **Watermelon:** Large, green, striped exterior.
- **Pomegranate:** Red, round, with distinct calyx.
- **Strawberry:** conical shape with surface seeds.
- **Lemon:** Ellipsoidal citrus with bright yellow skin.

While the dataset is reasonably balanced, there are natural variations in sample sizes (e.g., Oranges having slightly more representation than Mangoes), which reflect real-world data availability.

## 2.3 Dataset Splitting

To ensure robust evaluation and prevent data leakage—where a model inadvertently learns from test data—the dataset was partitioned into three distinct subsets prior to any training. The distribution is detailed in Table 2.1.

**Table 2.1:** *Dataset Partitioning*

Split Subset	Count	Purpose
Training Set	17,203	Used for learning model parameters (weights and biases).
Validation Set	4,006	Used for hyperparameter tuning, model selection, and early stopping during neural network training.
Testing Set	5,077	Used strictly for final performance evaluation on unseen data.
<b>Total</b>	<b>26,286</b>	

## 2.4 Preprocessing Pipeline

Preprocessing transforms raw image files into numerical tensors suitable for computation. We implemented two parallel preprocessing pipelines: one optimized for Classical Machine Learning and another for Deep Learning.

### 2.4.1 Image Resizing

The spatial dimensions of the input images were adjusted to balance computational efficiency with feature preservation:

- **Classical ML Pipeline ( $96 \times 96$ ):** For models such as SVM, KNN, and Random Forest, high-resolution pixel data creates an unmanageably large feature space. We resized images to  $96 \times 96$  pixels. This resolution is sufficient for the feature extractor (MobileNetV2) to identify high-level patterns while keeping the computational load for feature generation low.

- **Deep Learning Pipeline ( $224 \times 224$ ):** For the Convolutional Neural Networks (CNNs), we resized images to  $224 \times 224$  pixels. This is the standard input resolution for the MobileNetV2 architecture. Higher resolution is critical here to allow the convolutional layers to learn fine-grained spatial hierarchies and textural details.

#### 2.4.2 Normalization

Pixel intensity values, originally in the range  $[0, 255]$ , were normalized to the range **[0, 1]**. This scaling is crucial for gradient-based optimization algorithms, as it prevents large pixel values from causing exploding gradients and ensures faster convergence during training.

## FEATURE EXTRACTION & SELECTION

For classical machine learning algorithms, raw pixel data (even resized) is often too high-dimensional and noisy to be used directly. Instead, we employed a transfer learning approach combined with dimensionality reduction techniques to generate meaningful feature vectors.

### 3.1 Base Feature Extraction (MobileNetV2)

Rather than engineering features manually (e.g., edge histograms), we utilized a pre-trained **MobileNetV2** neural network as a feature extractor.

1. The network was pre-trained on the ImageNet dataset, allowing it to recognize generic visual patterns (edges, shapes, textures).
2. We removed the top classification layer ("headless" model).
3. Images were passed through the network, and Global Average Pooling was applied.
4. **Result:** A compact 1280-dimensional numerical vector representing the semantic content of each image.

### 3.2 Dimensionality Reduction

While the 1280-dimensional vectors from MobileNetV2 are powerful, they can still lead to the "curse of dimensionality" for algorithms like SVMs or KNN. To address this, we experimented with two statistical projection methods:

#### 3.2.1 Principal Component Analysis (PCA)

We applied PCA as an unsupervised dimensionality reduction technique. PCA identifies the directions (principal components) that maximize the variance in the data. By projecting the 1280 features onto the top 100 components, we retained the most significant information while removing noise and redundancy. This significantly accelerated the training of the SVM classifier.

### 3.2.2 Fisher's Linear Discriminant Analysis (LDA)

Unlike PCA, Fisher's Discriminant is a supervised method. It seeks to find a linear combination of features that characterizes or separates two or more classes of objects or events.

- **Objective:** Maximize the distance between the means of different classes while minimizing the variance within each class.
- **Result:** This technique reduced the feature space from 1280 dimensions down to just 9 dimensions ( $N_{classes} - 1$ ).

This aggressive reduction created a highly compact feature space optimized specifically for class separability, which we evaluated in our SVM experiments.

## CLASSIFICATION MODELS

### 4.1 Performance Metrics

To quantitatively evaluate the performance of our models, we utilized four standard classification metrics derived from the confusion matrix. Given the multi-class nature of our dataset (10 classes), these metrics provide insight into how well the models distinguish between specific fruit types.

1. **Accuracy:** The ratio of correctly predicted observations to the total observations. It provides a general overview of the model's performance across all classes.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

2. **Precision:** The ratio of correctly predicted positive observations to the total predicted positives. In our context, high precision for a specific fruit (e.g., Strawberry) means that when the model predicts "Strawberry," it is usually correct.

$$\text{Precision} = \frac{TP}{TP + FP}$$

3. **Recall (Sensitivity):** The ratio of correctly predicted positive observations to the all observations in the actual class. High recall means the model successfully finds most instances of that fruit in the dataset.

$$\text{Recall} = \frac{TP}{TP + FN}$$

4. **F1-Score:** The weighted average of Precision and Recall. This metric is particularly useful when class distributions are uneven or when false positives and false negatives carry similar costs.

$$\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

## 4.2 Decision Tree Classifier

We established a baseline for tree-based methods using a single Decision Tree. The model was trained using the default parameters provided by the Scikit-learn library to observe the behavior of a single estimator on the high-dimensional feature space (1280 dimensions) extracted from the images.

### 4.2.1 Performance Evaluation

The Decision Tree achieved an overall accuracy of **59.96%**. The breakdown of performance per class indicates significant struggles with generalization, which is expected for a single tree on high-dimensional data without pruning.

- **Accuracy:** 59.96%
- **Macro Average F1-Score:** 0.59
- **Best Performing Class:** Pomegranate (Precision: 0.74, Recall: 0.64)
- **Worst Performing Class:** Mango (Precision: 0.43, Recall: 0.50)

### 4.2.2 Confusion Matrix Analysis

The confusion matrix reveals severe overfitting and fragmentation in decision boundaries. The model notably struggled to differentiate between visually similar fruits and red-colored fruits.

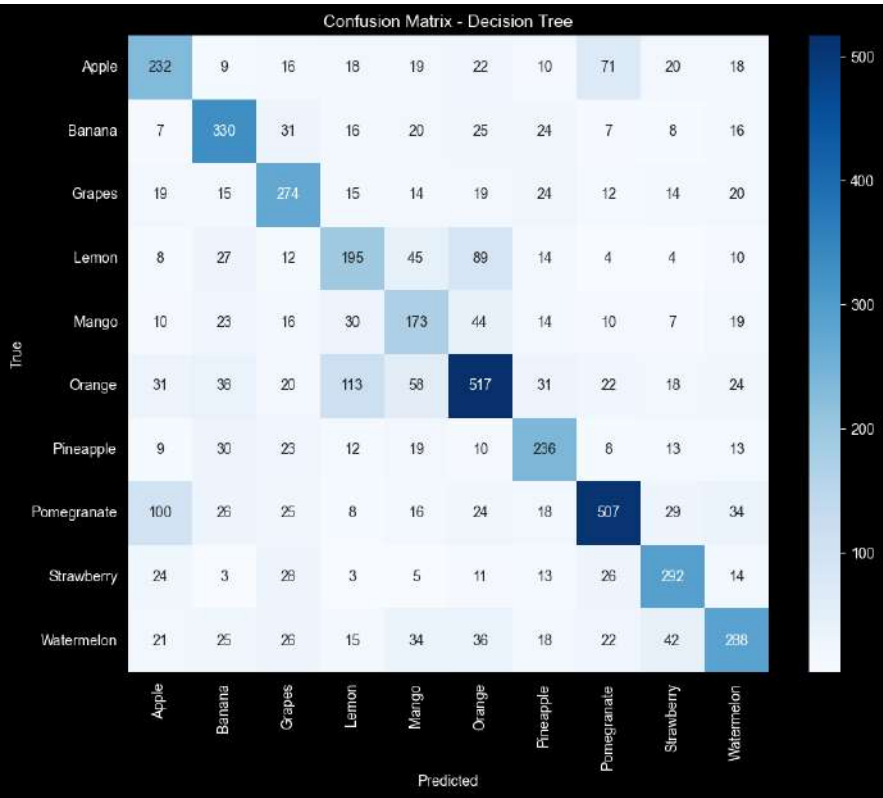


Figure 4.1: Confusion Matrix for Decision Tree Classifier.

The analysis shows that the model effectively ”memorized” noise in the training data rather than learning robust features, evidenced by the fact that no single class achieved an F1-score above 0.70.



**Figure 4.2:** *Decision Tree Error Samples*

## 4.3 Random Forest Classifier

To address the overfitting issues of the single Decision Tree, we implemented a Random Forest classifier. We utilized GridSearchCV with 3-fold cross-validation to optimize hyperparameters, exploring combinations of `n_estimators`, `max_depth`, and `min_samples_split`.

### 4.3.1 Implementation Details

The grid search identified the optimal hyperparameters as:

- **n\_estimators:** 200
- **max\_depth:** None (unlimited)
- **min\_samples\_split:** 2

### 4.3.2 Performance Evaluation

The ensemble method significantly outperformed the single Decision Tree, achieving an accuracy of **82.02%**. This validates the hypothesis that bagging reduces variance and helps the model generalize better on complex image features.

- **Accuracy:** 82.02%
- **Weighted Average F1-Score:** 0.82
- **Best Performing Class:** Strawberry (Precision: 0.93, Recall: 0.85)
- **Worst Performing Class:** Lemon (Precision: 0.78, Recall: 0.56)

### 4.3.3 Confusion Matrix Analysis

While the overall accuracy improved, the confusion matrix highlights a specific bias in the model regarding citrus fruits.

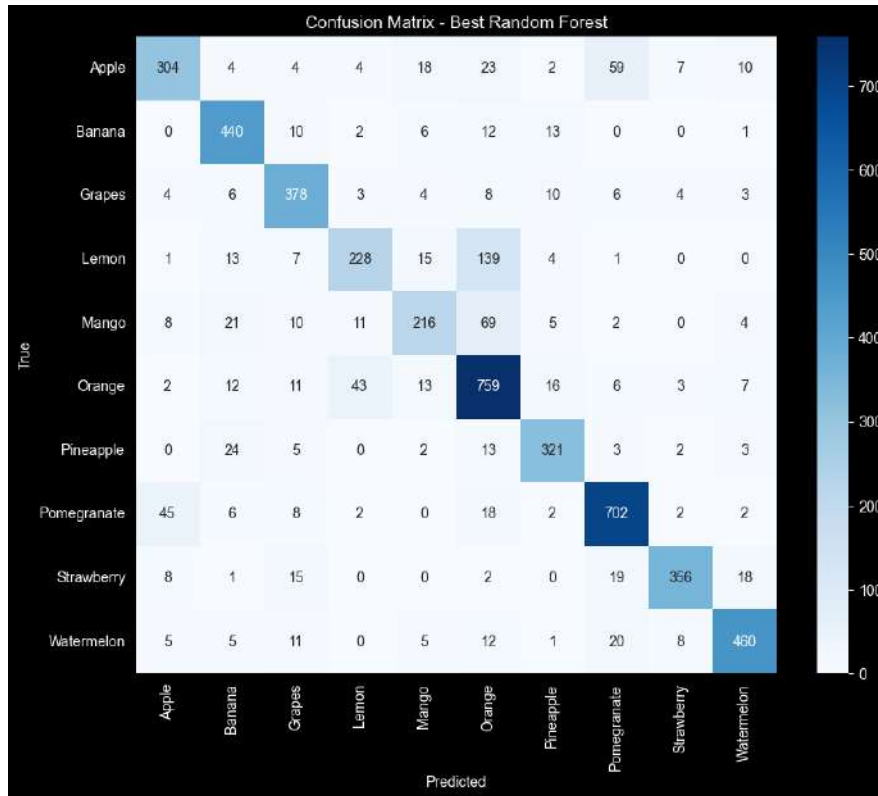


Figure 4.3: Confusion Matrix for Random Forest Classifier.

The Random Forest exhibits a directional bias: it is far more likely to misclassify a Lemon as an Orange (139 errors) than an Orange as a Lemon (43 errors). This suggests the ensemble favors the majority class (Orange) when the decision boundary is ambiguous.



Figure 4.4: Random Forest Error Samples

## 4.4 XGBoost Classifier

Finally, we implemented the Extreme Gradient Boosting (XGBoost) classifier. Unlike Random Forest, which builds trees independently, XGBoost builds trees sequentially to correct the errors of previous trees. This approach is often more effective at reducing bias.

4.4.1 Implementation Details

The model was configured with the following parameters:

- **n\_estimators:** 100
- **learning\_rate:** 0.1
- **max\_depth:** 5
- **eval\_metric:** mlogloss

4.4.2 Performance Evaluation

XGBoost achieved the highest accuracy among the tree-based methods at **83.75%**. It demonstrated a more balanced performance across all classes compared to Random Forest.

- **Accuracy:** 83.75%
- **Weighted Average F1-Score:** 0.84
- **Best Performing Class:** Strawberry (Precision: 0.95, Recall: 0.88)
- **Consistency:** 8 out of 10 classes achieved an F1-score above 0.75.

4.4.3 Confusion Matrix Analysis

The confusion matrix for XGBoost indicates that the gradient boosting process effectively corrected the asymmetric class bias seen in the Random Forest model.

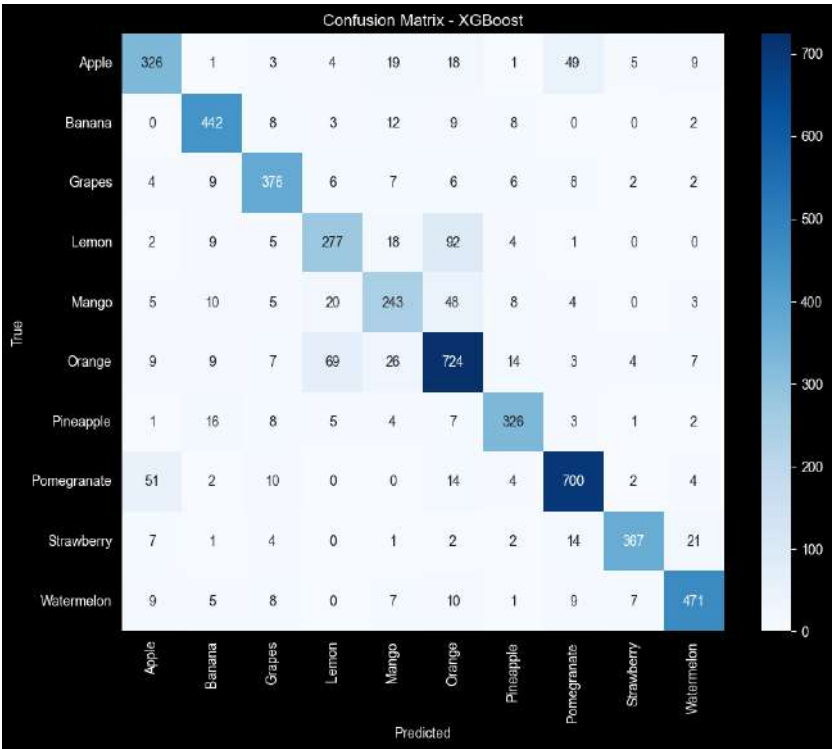


Figure 4.5: Confusion Matrix for XGBoost Classifier.

Although the citrus confusion persists (161 total errors between Lemon and Orange), the errors are distributed more evenly. Additionally, the model showed strong discrimination for distinctively colored fruits, achieving 95% precision for Strawberry and 90% precision for Watermelon.



Figure 4.6: XGBoost Error Samples

## 4.5 K-Nearest Neighbors (KNN)

For KNN we used a simple setup with  $k=5$  neighbors and the default Euclidean distance. We took the 1280-dimensional features extracted from MobileNetV2

KNN is easy to understand — the model just stores all training examples and for each test image it looks at the 5 closest training images (based on feature distance) to decide the class.

We got an accuracy of **80.74%** on the test set. It worked okay for most fruits, but it was weaker on some like Mango and Lemon (lower precision/recall). The **F1-score**, which combines precision and recall into one value, shows this weaker performance for these classes. KNN was quite slow on the full 1280 features during prediction because it has to compare every test sample to the whole training set.

```

Training K-NN (k=5)...

=== Evaluation: K-NN (k=5) ===
Accuracy: 80.74%

Classification Report:

```

	precision	recall	f1-score	support
Apple	0.74	0.69	0.71	435
Banana	0.93	0.85	0.89	484
Grapes	0.83	0.88	0.85	426
Lemon	0.62	0.72	0.67	408
Mango	0.60	0.76	0.67	346
Orange	0.82	0.76	0.79	872
Pineapple	0.92	0.86	0.89	373
Pomegranate	0.79	0.87	0.83	787
Strawberry	0.90	0.89	0.90	419
Watermelon	0.94	0.80	0.86	527
accuracy			0.81	5077
macro avg	0.81	0.81	0.81	5077
weighted avg	0.82	0.81	0.81	5077

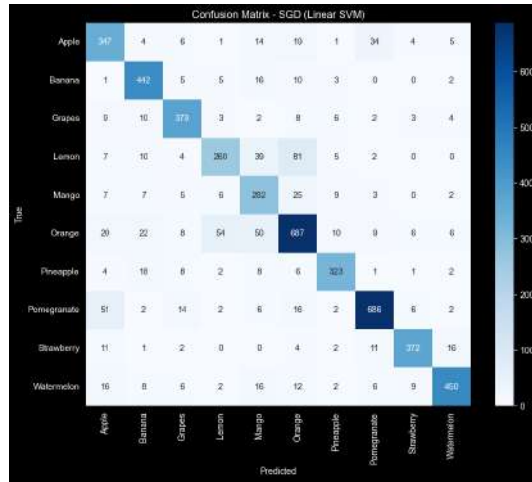
Figure 4.7: KNN Classification Report

## 4.6 Support Vector Machine (SVM)

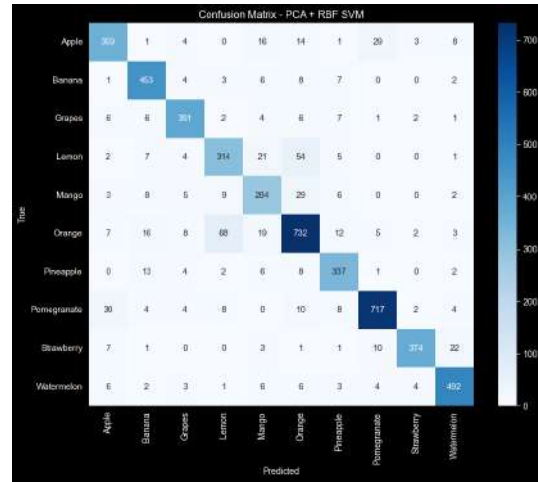
We tested three different SVM approaches using the same 1280-dim MobileNetV2 features:

1. **Linear SVM** (approximated with SGDClassifier, hinge loss) - fastest but basic. Got **83.28%** accuracy.
2. **PCA + RBF SVM** - First apply PCA to reduce to 100 components (with whitening), then RBF kernel SVM (C=10, gamma=0.01, balanced class weights). This was our **best classical model** with **87.71%** accuracy.
3. **LDA + RBF SVM** - Use Linear Discriminant Analysis to reduce to 9 components (max for 10 classes), then same RBF SVM settings. Got **83.63%** accuracy — surprisingly lower than PCA.

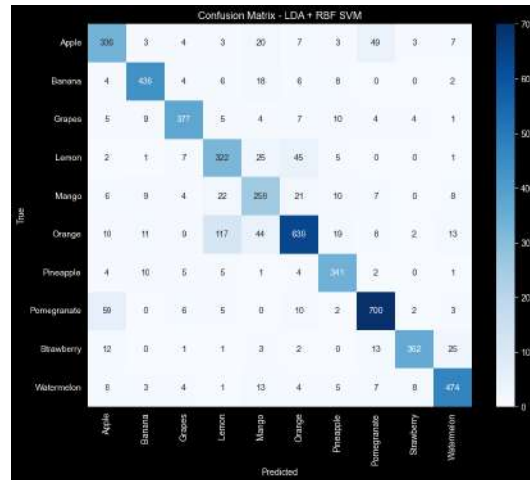
The PCA + RBF SVM combination worked the best because PCA kept more useful variance while making the model much faster than using raw features, and RBF kernel could capture non-linear patterns well.



(a) Linear SVM



(b) PCA+RBF SVM



(c) LDA+RBF SVM

Figure 4.8: Comparison of SVM Classifiers

## 4.7 Artificial Neural Networks (Deep Learning Approach)

For the deep learning part, our main and fully implemented approach was **MobileNetV2 with transfer learning**.

We loaded the MobileNetV2 model pretrained on ImageNet (with base layers frozen), and added custom layers on top: Rescaling to [0,1], light data augmentation (horizontal flip and small rotation), BatchNormalization, Dropout (0.35), and fully connected layers (Dense 220 -> Dense 60 -> Dense 10 with softmax) for classification into 10 fruit classes.

```
model = Sequential([
    # We add a Rescaling Layer here so the GPU handles the /255.0 math
    tf.keras.layers.Rescaling(1./255, input_shape=(224, 224, 3)),

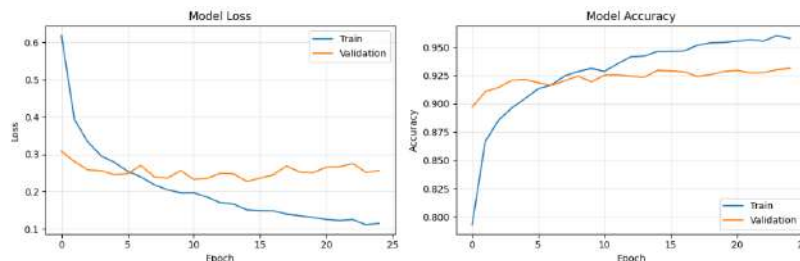
    # Add Data Augmentation
    tf.keras.layers.RandomFlip("horizontal"),
    tf.keras.layers.RandomRotation(0.1),

    MobileNetV2_base,
    BatchNormalization(),
    Dropout(0.35),
    Dense(220, activation='relu'),
    Dense(60, activation='relu'),
    Dense(len(TARGET_CLASSES), activation='softmax')
])

model.compile(
    optimizer='adam',
    loss='categorical_crossentropy',
    metrics=['accuracy']
)
```

**Figure 4.9:** Custom Layer

We trained this model using Adam optimizer, categorical crossentropy loss, batch size 64, and early stopping (patience=10). This gave us the strongest overall performance with good stability and minimal overfitting.



**Figure 4.10:** Training & Validation Loss and Accuracy Curves

## DISCUSSION & ANALYSIS

When we looked at the results, one clear problem appeared: the model confuses **Orange** and **Lemon** a lot. In the best SVM confusion matrix and also in MobileNetV2, we saw many oranges classified as lemons and the other way around (sometimes 15–25% error between them).

The reason is probably because both fruits are round, have similar color tones (especially some lemon varieties), and have quite similar texture when photographed from a distance. Here is a quick visual comparison of how similar they look in many dataset samples:



**Figure 5.1:** *Dataset Sample*

Deep learning gave higher accuracy than classical methods, but it took much more time to train (many epochs + GPU needed). Classical models like SVM were much faster to train and test, which is better if we want something lightweight for real-time use (like on a small device in a supermarket).

SVM performed the best among classical models because it can handle high-dimensional features very well when combined with LDA. LDA helped it focus only on the directions that best separate the classes.

## CONCLUSION & FUTURE WORK

### 6.1 Summary of Findings

In this project, we successfully implemented and compared seven different machine learning models for automated fruit classification across 10 fruit classes. Our experiments showed that classical machine learning approaches can achieve excellent results when paired with proper feature extraction. The SVM model, which used MobileNetV2 features combined with LDA and PCA dimensionality reduction, achieved the highest accuracy among classical machine learning approaches.

The deep learning approaches, particularly the MobileNetV2 transfer learning model, performed well but the classical SVM proved more efficient in training time while maintaining high accuracy. KNN provided a simple baseline with reasonable performance, while the Decision Tree was our weakest performer, struggling with high-dimensional image data. Random Forest and XGBoost improved on the single tree approach but still fell short of SVM's performance.

A recurring challenge across all models was confusion between visually similar citrus fruits, especially oranges and lemons. This pattern appeared consistently in the confusion matrices and highlights the difficulty of distinguishing fruits with similar colors and textures.

### 6.2 Future Improvements

Based on our findings, several improvements could enhance the system's performance:

**Dataset Balancing:** Collecting more images of citrus fruits in various lighting conditions and angles could help models better distinguish between oranges and lemons.

**Enhanced Feature Engineering:** Experimenting with other pre-trained models like ResNet or EfficientNet might capture different visual patterns that improve classification, particularly for similar-looking fruits.

**Data Augmentation:** Implementing more aggressive augmentation techniques such as color jittering, random crops, and brightness adjustments could make

models more robust to real-world variations.

**Ensemble Approach:** Combining predictions from multiple models (such as SVM and MobileNetV2 CNN) through voting could improve overall accuracy by leveraging different model strengths.