CMPE 462 - Machine Learning: Assignment 1

Asaf Kanlipicak, Ali Sonmez, Saur

November 28, 2025

1 Introduction

This report outlines our work on developing a multi-class classification model for fruit and vegetables. For the project, we needed to create a custom dataset consisting of five distinct classes: banana, carrot, cucumber, mandarin, and tomato. Two primary goals of the project was to understand the mechanics of classification algorithms by building one from scratch and facing with the difficulties of data collection.

In the following sections, we describe how we collected and processed our data (combining images, text, and numerical attributes), detail our custom implementation, and compare its performance against the standard Scikit-learn library.

2 Dataset

Examples

Our dataset includes **5922** images in total. We have **301** real photographs and **5621** generated images.

Table 1: Distribution of Generated and Photograph Images per Class

Class	**Generated**	**Photograph**	**Total**
Banana	1000	100	1100
Carrot	1173	50	1223
Cucumber	1000	50	1050
Mandarin	1000	51	1051
Tomato	1448	50	1498
Total	**5621**	**301**	**5922**

We have separated 0.8 of the generated images as our **training** set and 0.2 of it as our **validation** set. To see if a model which is trained on generated images can predict real photographs, we have selected all the photographs as our **test** set. Below are a few examples.

Figure 1: A generated banana

Table 2: Example Feature Vector (Class: Banana)

Feature Group	**Attributes**	**Values**
Grayscale Histogram	gray_000 . . . gray_005 . . . gray_063	2.0, 2.0, 13.0, 76.0, 77.0, 51.0 . . . . . . 205.0
Color Statistics	blue_mean, blue_std green_mean, green_std red_mean, red_std	73.93, 67.23 133.87, 84.41 163.57, 100.53
Physical Attributes	weight, size	130.93 g, 17.86 cm
Text Description	description	"soft yellow"
Target Label	**class**	**banana**

Figure 2: A generated carrot

Table 3: Example Feature Vector (Class: Carrot)

Feature Group	**Attributes**	**Values**
Grayscale Histogram	gray_000 … gray_005 … gray_063	221.0, 217.0, 208.0, 193.0, 206.0, 202.0 … … 172.0
Color Statistics	blue_mean, blue_std green_mean, green_std red_mean, red_std	172.12, 57.35 178.13, 48.76 190.24, 35.87
Physical Attributes	weight, size	46.06 g, 11.84 cm
Text Description	description	"orange temperate"
Target Label	**class**	**carrot**

Figure 3: A generated cucumber

Table 4: Example Feature Vector (Class: Cucumber)

Feature Group	**Attributes**	**Values**
Grayscale Histogram	gray_000 … gray_005 … gray_063	175.0, 168.0, 163.0, 159.0, 160.0, 154.0 … … 190.0
Color Statistics	blue_mean, blue_std green_mean, green_std red_mean, red_std	149.94, 82.32 173.52, 66.73 175.36, 70.31
Physical Attributes	weight, size	287.30 g, 25.95 cm
Text Description	description	"temperate long"
Target Label	**class**	**cucumber**

Figure 4: A generated mandarin

Table 5: Example Feature Vector (Class: Mandarin)

Feature Group	**Attributes**	**Values**
Grayscale Histogram	gray_000 . . . gray_005 . . . gray_063	186.0, 204.0, 225.0, 234.0, 230.0, 200.0 . . . . . . 110.0
Color Statistics	blue_mean, blue_std green_mean, green_std red_mean, red_std	164.11, 85.86 190.25, 53.10 212.07, 44.73
Physical Attributes	weight, size	76.19 g, 7.87 cm
Text Description	description	"sour soft"
Target Label	**class**	**mandarin**

Figure 5: A generated tomato

Table 6: Example Feature Vector (Class: Tomato)

Feature Group	**Attributes**	**Values**
Grayscale Histogram	gray_000 … gray_005 … gray_063	135.0, 144.0, 122.0, 147.0, 218.0, 159.0 … … 50.0
Color Statistics	blue_mean, blue_std green_mean, green_std red_mean, red_std	100.99, 72.33 112.19, 72.33 130.97, 63.93
Physical Attributes	weight, size	108.11 g, 4.12 cm
Text Description	description	"soft sour"
Target Label	**class**	**tomato**

Data Collection

To obtain a large dataset we employed two techniques:

- **Generation:** The majority of the data was generated using **Stability AI's Stable Diffusion 3.5 Medium** model. Our prompts are visible in the generation notebook.
- **Manual Photography:** Additionally, we manually took photographs of the actual fruits and vegetables. These are used as the test set.

Preprocessing Pipeline

We wrote a preprocessing script (`image_processing.ipynb`) to standardize the raw inputs before feature extraction. Our pipeline is:

1. **Resizing:** All images, regardless of their original resolution, are resized to a uniform dimension of 512×512 **pixels**.
2. **Randomization:** A fixed seed (`SEED = 462`) was used to shuffle the dataset for reproducability concerns.

Dataset Splitting Strategy

We separated the data as follows:

- **Training & Validation Sets:** Derived from the **generated** images. The generated data was shuffled and split with an **80/20 ratio**:
  - **80%** allocated to the Training set.
  - **20%** allocated to the Validation set.
- **Test Set:** Composed only of the **real-world photographs**.

With the help of this splitting, we can test an interesting case in which a model is trained using generated images, but tested on the real ones.

Feature Extraction

We have **73** features in total, 64 of which is the grayscale representation of the image. The other 8 is related to colors, physical attributes, and semantic information.

Grayscale Histogram (8 $\times$ 8 = 64 features)

We convert each image into a 8$\times$8 grayscale image and give the value of each pixel as a feature to our model. This helps distinguishing between objects with similar colors but different surface textures (e.g., the smooth surface of a tomato vs. the rougher texture of a carrot).

How did we extract: We made use of OpenCV to convert the image into grayscale and resize it to 8$\times$8.

Color Statistics (RGB Mean and Standard Deviation)

Color is the most intuitive discriminator for this specific dataset.

- **Mean:** Captures the dominant color of the object. This is crucial to separate the objects with different colors such as banana and mandarin.
- **Standard Deviation:** Captures the color variance. Even though we are not sure, we think that standard deviation might help the model learn some texture specific relation. (On the surface of a cucumber, there might be frequent dark spots.)

How did we extract: We made use of OpenCV to read the color value of each pixel in an image. Afterwards, it was just simple math to calculate the features.

Physical Attributes (Weight and Size)

Weight and size are important features since these help the model to learn, hopefully, the geometric attributes of an object.

- These features define the geometric and physical boundaries of the classes (A cucumber is significantly longer/larger than a mandarin).

How did we extract: This might be the trickiest one to extract. We used Gaussian distributions with different mean and variance values for each class to generate weights and sizes. For example, the average weight of a banana is 120 g, with a variance of 16 g^2 in the code. These values were derived by asking to ChatGPT since there are multiple answers on the Internet, but we want one.

Semantic Features (Text Description)

We have selected a set of words to define different classes. However, we paid attention not to make them easily separable. For example, if an object can have the word “tropical” in its text feature, then, another one also can.

- We aim to make the model learn the relation between a *bag of words* and a class. With combination of words, it is feasible to separate the objects that share some common words.

How did we extract: We created a vocabulary in which we aggregated all unique descriptors (e.g., textures, climates) across all classes. We then represented the text description of each sample as a binary feature vector, where each dimension corresponds to the presence (1) or absence (0) of a specific word from this vocabulary. This resulted in a representation whcih encodes the attributes defined in our vocabulary.

To integrate the multi-modal data, we concatenated the normalized feature vectors from the visual, physical, and semantic domains into a single unified vector (in a csv file).

3 Logistic Regression Implementation

We have implemented the logistic regression using `numpy` and run tests comparing with `Scikit-learn`.

Only Image

In this experiment, we trained the model using only the flattened grayscale histograms and color statistics (Visual Features).

Performance Metrics

The below table shows the overall metrics. While the model achieved an F1-score of **74.16%** on the validation set (generated images), performance dropped to **32.12%** on the test set (real photographs). This suggests the model learned visual features specific to the generative process and did not generalize to real-world noise.

Table 7: Performance Metrics for Image-Only Classifier

Metric	**Training**	**Validation**	**Test (Photos)**
Accuracy	77.20%	75.11%	32.89%
Precision	0.7652	0.7434	0.4359
Recall	0.7665	0.7483	0.3607
F1 Score	0.7624	0.7416	0.3212

Class-wise AUC Analysis

The Area Under the Curve (AUC) scores in Table 8 shows class difficulties.

- **Carrot** was the hardest class consistently (Train AUC 0.77, Test AUC 0.56).
- **Mandarin** showed the best generalization, maintaining a relatively high AUC of 0.81 on real photographs compared to other classes.

Table 8: Class-wise AUC Scores (One-vs-All)

Class	Train AUC	Val AUC	Test AUC
Banana	0.9434	0.9343	0.7039
Carrot	0.7761	0.7424	0.5610
Cucumber	0.9736	0.9657	0.6414
Mandarin	0.9000	0.9135	0.8111
Tomato	0.9767	0.9719	0.7428

Training Visualizations

Figures 6 and 7 display the training dynamics. The loss curve shows stable convergence without significant overfitting on the synthetic data.

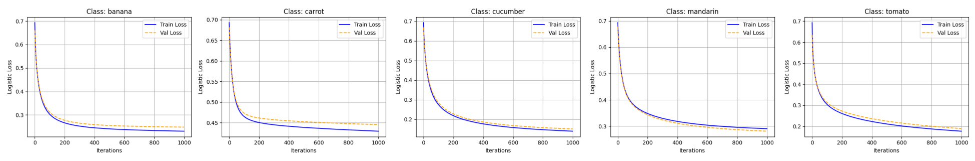

Figure 6: Training and Validation Loss during the optimization process.

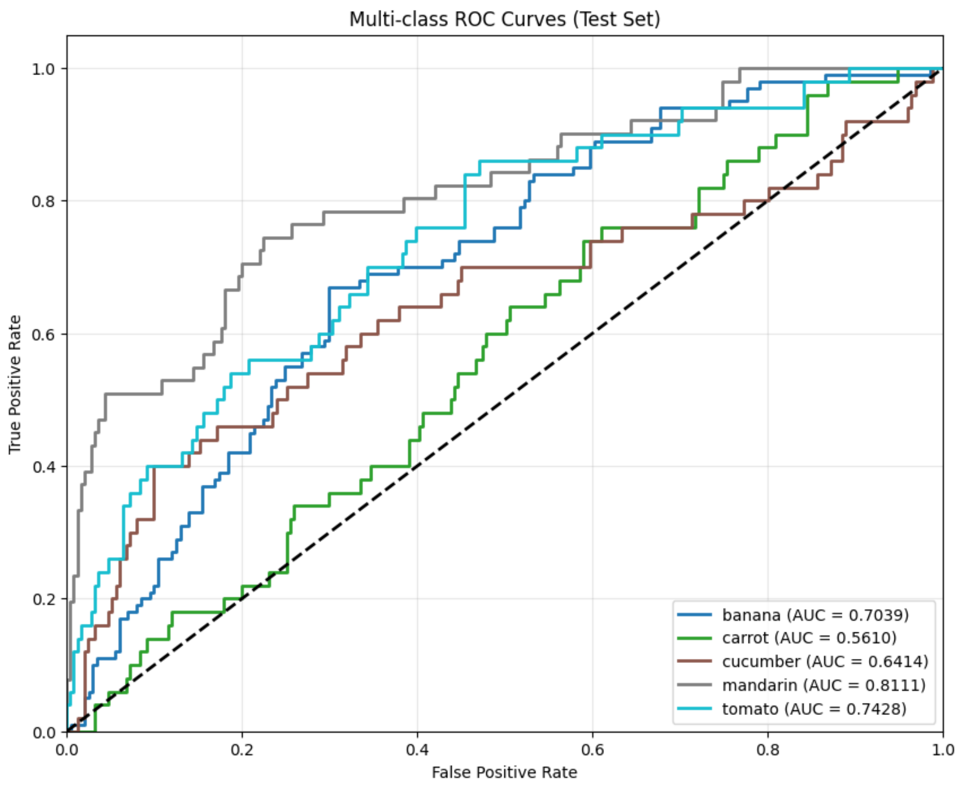

Figure 7: ROC Curves evaluating model performance on the Test Set (Photographs).

Only Numeric

For this experiment, we utilized only the physical attributes (Weight and Size). These features represent a low-dimensional input space compared to images or text.

Performance Metrics

As shown in Table 9, the model achieves moderate accuracy. The performance is consistent across Training, Validation, and Test sets. Unlike the image modality, the physical attributes generalize well to real-world data, likely because weight and size attributes are generated in the same way. However, overall results show that these two features alone are not enough for a good classification.

Table 9: Performance Metrics for Numeric-Only Classifier

Metric	**Training**	**Validation**	**Test (Photos)**
Accuracy	60.01%	62.40%	60.80%
Precision	0.5245	0.5865	0.6614
Recall	0.5585	0.5831	0.5858
F1 Score	0.5054	0.5293	0.5161

Class-wise AUC Analysis

The AUC scores (Table 10) indicate that certain classes are easily separable by looking at their physical attributes.

- **Carrot and Banana** have high AUC scores (> 0.90), suggesting their length-to-weight ratios are unique.
- **Cucumber** proved difficult on the test set, possibly due to high variance in real-world cucumber sizes.

Table 10: Class-wise AUC Scores (One-vs-All) - Numeric Only

Class	**Train AUC**	**Val AUC**	**Test AUC**
Banana	0.9087	0.9203	0.9060
Carrot	0.9289	0.9212	0.9249
Cucumber	0.8391	0.8452	0.7768
Mandarin	0.8919	0.8825	0.9492
Tomato	0.8442	0.8593	0.8556

Training Visualizations

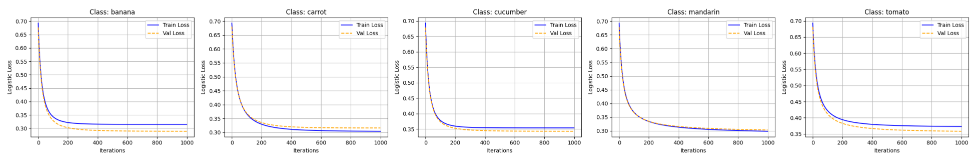

Figure 8: Training and Validation Loss (Numeric-Only).

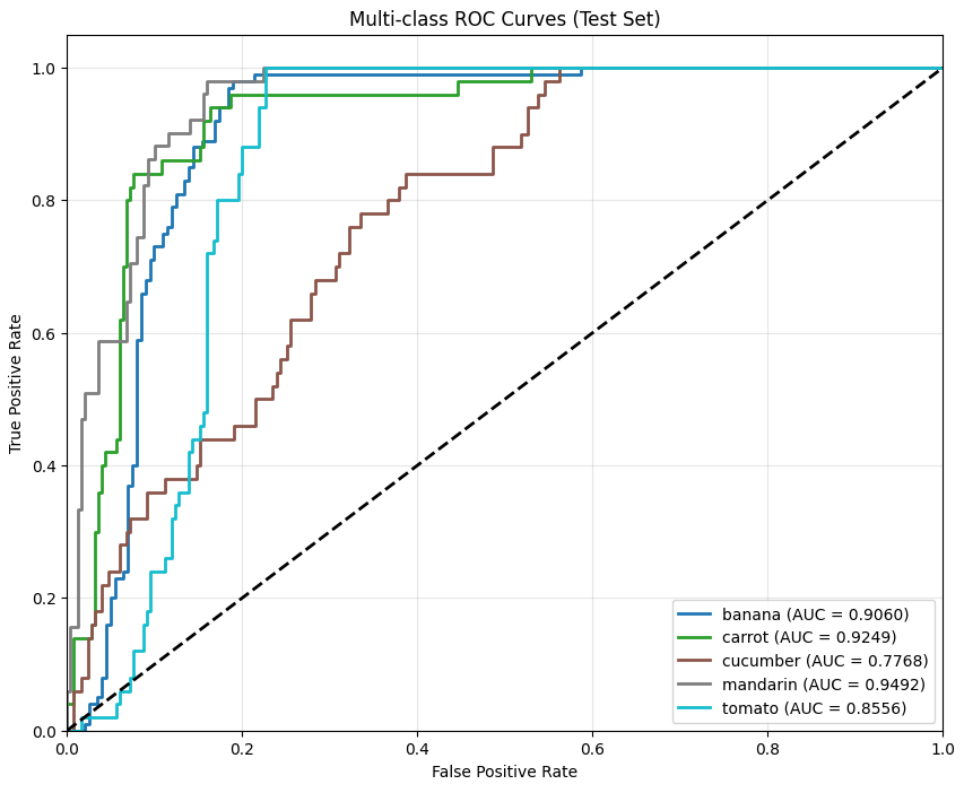

Figure 9: ROC Curves on Test Set (Numeric-Only).

Only Text

This classifier was trained using binary feature vectors derived from the vocabulary of text descriptions.

Performance Metrics

These features (Table 11) outperformed both image and numeric ones, achieving approximately 80% accuracy on the validation set. Also, it maintained approximately 77% accuracy on the test set, showing nice generalization.

Table 11: Performance Metrics for Text-Only Classifier

Metric	**Training**	**Validation**	**Test (Photos)**
Accuracy	79.78%	80.00%	77.41%
Precision	0.8170	0.8146	0.8133
Recall	0.7828	0.7853	0.7537
F1 Score	0.7831	0.7875	0.7553

Class-wise AUC Analysis

AUC scores were consistently high (> 0.95) for all classes across all splits. This confirms that the semantic descriptions provided in the dataset are highly discriminative even though there is randomization and multiple words.

Table 12: Class-wise AUC Scores (One-vs-All) - Text Only

Class	**Train AUC**	**Val AUC**	**Test AUC**
Banana	0.9727	0.9692	0.9614
Carrot	0.9671	0.9651	0.9547
Cucumber	0.9759	0.9764	0.9749
Mandarin	0.9621	0.9652	0.9504
Tomato	0.9934	0.9947	0.9859

Training Visualizations

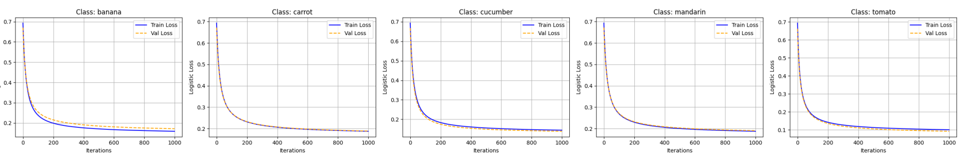

Figure 10: Training and Validation Loss (Text-Only).

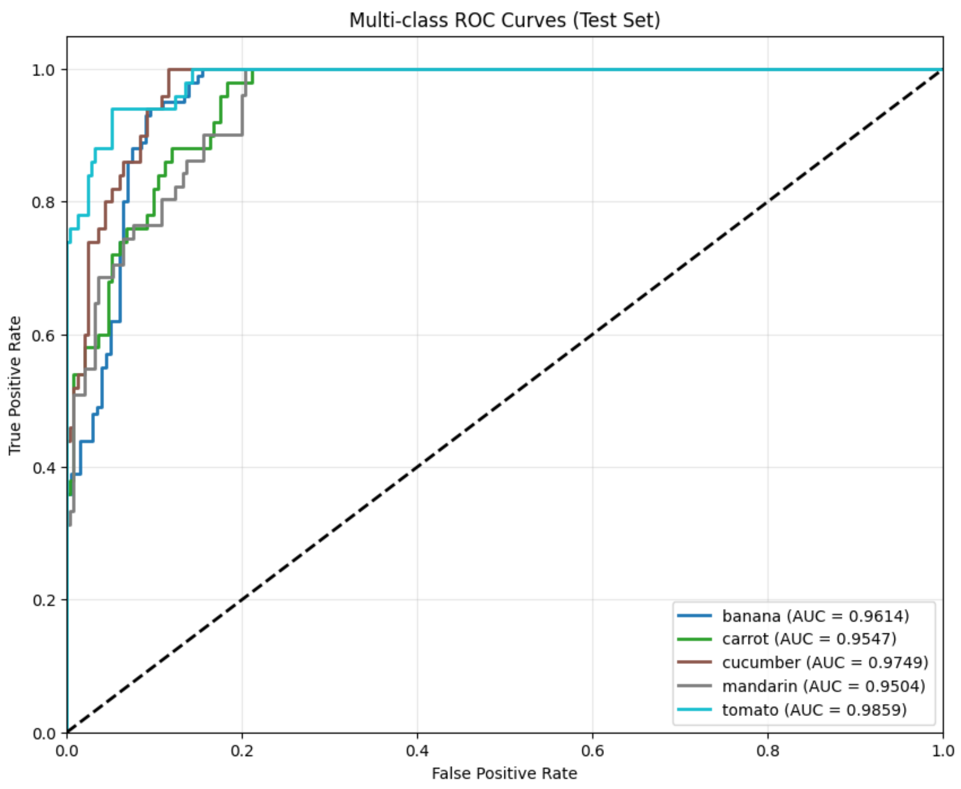

Figure 11: ROC Curves on Test Set (Text-Only).

Fused

Finally, we trained the classifier on the concatenated feature vector, combining Image, Numeric, and Text features.

Performance Metrics

The fused model achieved the best overall performance. As shown in Table 13, it reached near-perfect accuracy on the training and validation data. On the real-world test set, accuracy remained high at **88.37%**, outperforming the single-feature baselines (Image: 32%, Numeric: 60%, Text: 77%).

Table 13: Performance Metrics for Fused Data Classifier

Metric	**Training**	**Validation**	**Test (Photos)**
Accuracy	98.47%	98.04%	88.37%
Precision	0.9845	0.9795	0.8996
Recall	0.9837	0.9792	0.8705
F1 Score	0.9840	0.9793	0.8654

Class-wise AUC Analysis

The AUC scores are also well, with nearly all classes achieving > 0.98 on the test set, indicating the model classifies reliably.

Table 14: Class-wise AUC Scores (One-vs-All) - Fused Data

Class	Train AUC	Val AUC	Test AUC
Banana	0.9990	0.9989	0.9919
Carrot	0.9954	0.9903	0.9854
Cucumber	0.9993	0.9989	0.9799
Mandarin	0.9991	0.9996	0.9984
Tomato	0.9997	0.9994	0.9968

Training Visualizations

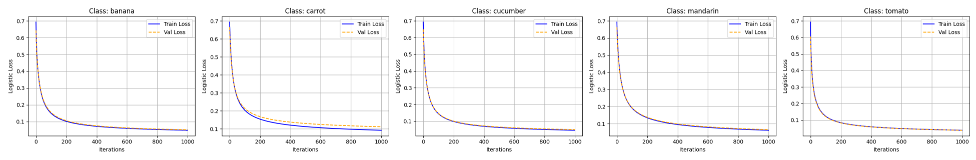

Figure 12: Training and Validation Loss (Fused Data).

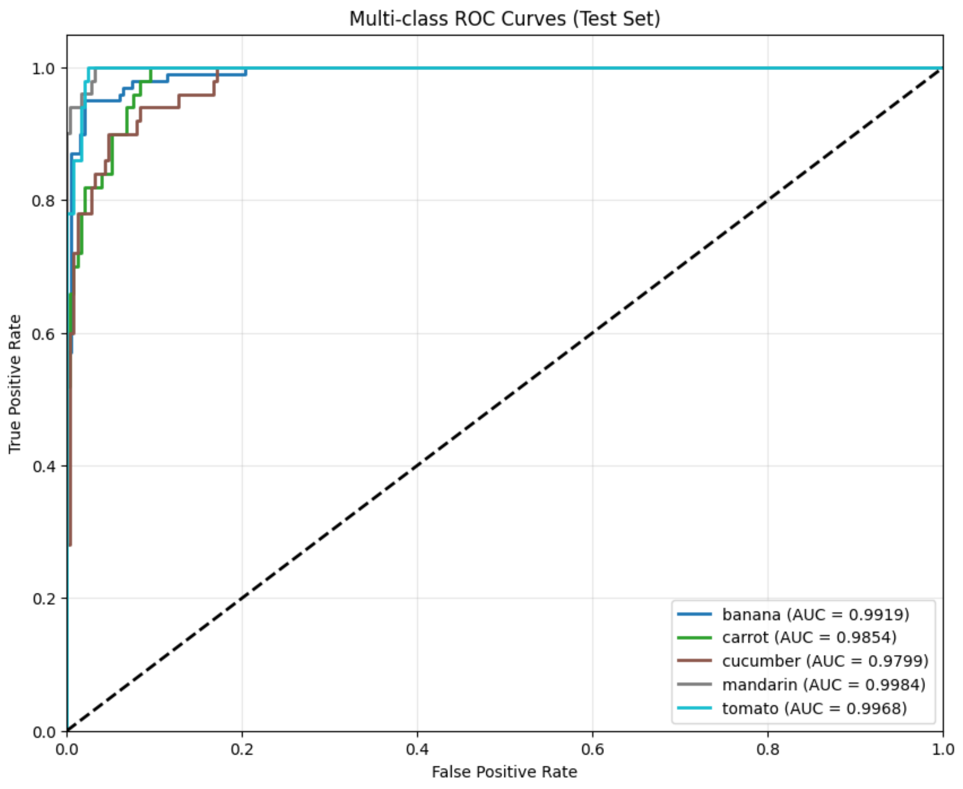

Figure 13: ROC Curves on Test Set (Fused Data).