Smart Basket for Automatic Product Classification and Counting

Project Report by Iuliia Bolgova

August 2024

1. Introduction

In modern supermarkets and grocery stores, the process of identifying and counting products often relies on manual input. This method is slow and prone to errors, leading to longer checkout times and incorrect pricing. These inefficiencies negatively affect both customer experience and store operations.

To address this, our project proposes the development of a "smart basket" that automatically recognizes products using image-based classification. This approach has the potential to significantly accelerate the shopping process by eliminating the need for manual scanning, thus improving accuracy and speed at checkout.

As the retail industry moves towards automation and efficiency, there is a growing demand for smart solutions that streamline the shopping experience. A smart basket, utilizing image processing to identify and count products, can eliminate the need for manual entry, making the checkout process faster and more accurate. This project leverages advanced image processing techniques to develop a model capable of classifying and counting products in real time.

2. Problem Definition

Problem Statement

The manual process of product identification and counting in stores leads to inefficiencies at checkout, increasing both time and the likelihood of errors. This project aims to automate the process using image-based classification, thereby speeding up the checkout process and reducing human error.

Context

Automation and efficiency are key drivers in modern retail. Implementing a smart basket with image classification technology to detect products and tally counts will streamline the shopping experience, allowing customers to check out faster while reducing errors in product identification.

Project Objectives

- **Develop a model** for automatic classification and counting of fruits and vegetables.
- **Use image processing techniques** to improve recognition accuracy.
- **Implement a system** that can be integrated into a smart basket for retail use.
-

3. Data

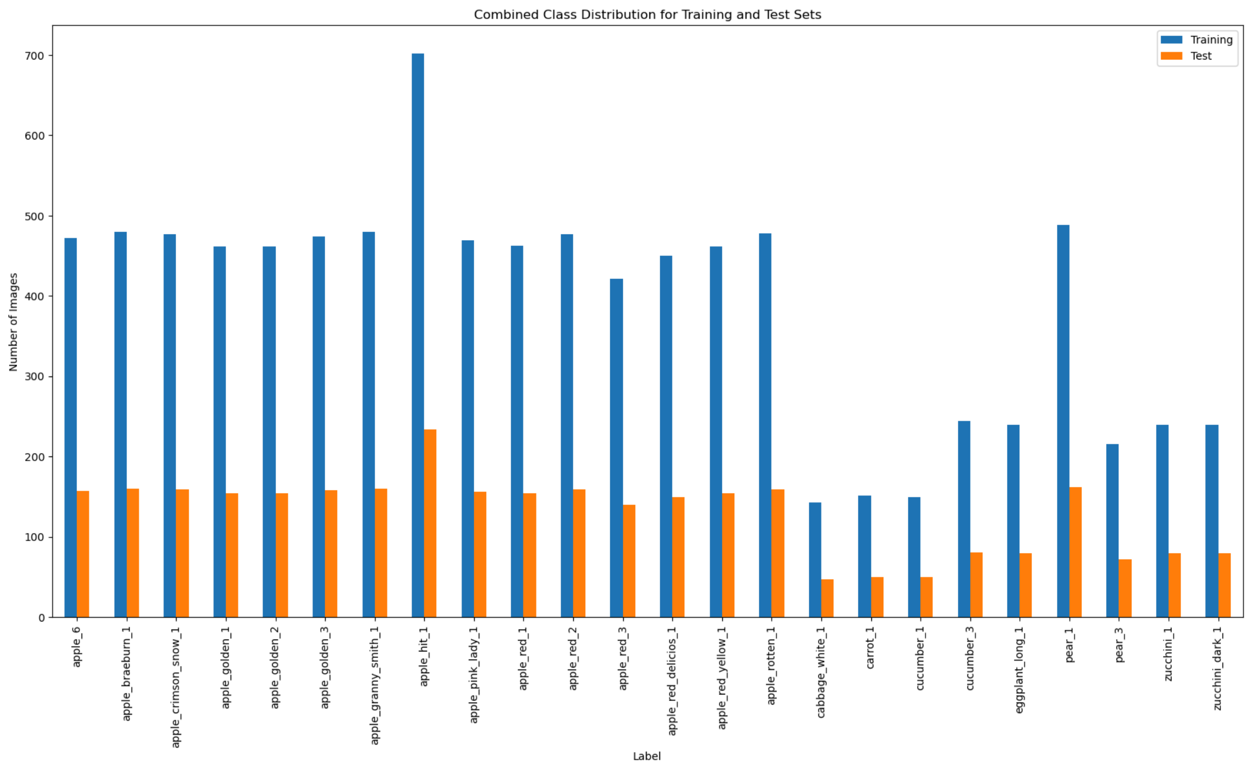
The Fruits-360 dataset was selected for both training and testing purposes. This dataset contains over 70,000 images of 120 different types of fruits and vegetables, making it a robust choice for image classification tasks, particularly in retail and agricultural sectors.

4. Exploratory Data Analysis

A detailed analysis of the dataset was conducted to explore the distribution of images across different categories of fruits and vegetables. The dataset was split into training, testing, and validation sets to ensure an appropriate evaluation of model performance. An assessment of class distributions across these splits provided insights into any imbalances that might affect the learning process.

Data Summary

The dataset consists of 24 unique labels representing different fruits and vegetables. Upon analysis, certain imbalances were observed in the distribution of images between the training and test sets. The figure below shows the class distribution of images across the training and test sets.



The blue bars in the chart represent the number of images in the training set, while the orange bars represent the test set. It is evident that some classes have a significantly higher representation than others, which might require data augmentation or sampling strategies during the training phase. For example, classes such as apple_ht_1 and pear_1 have a noticeable disparity in the number of training samples compared to others.

Key Observations

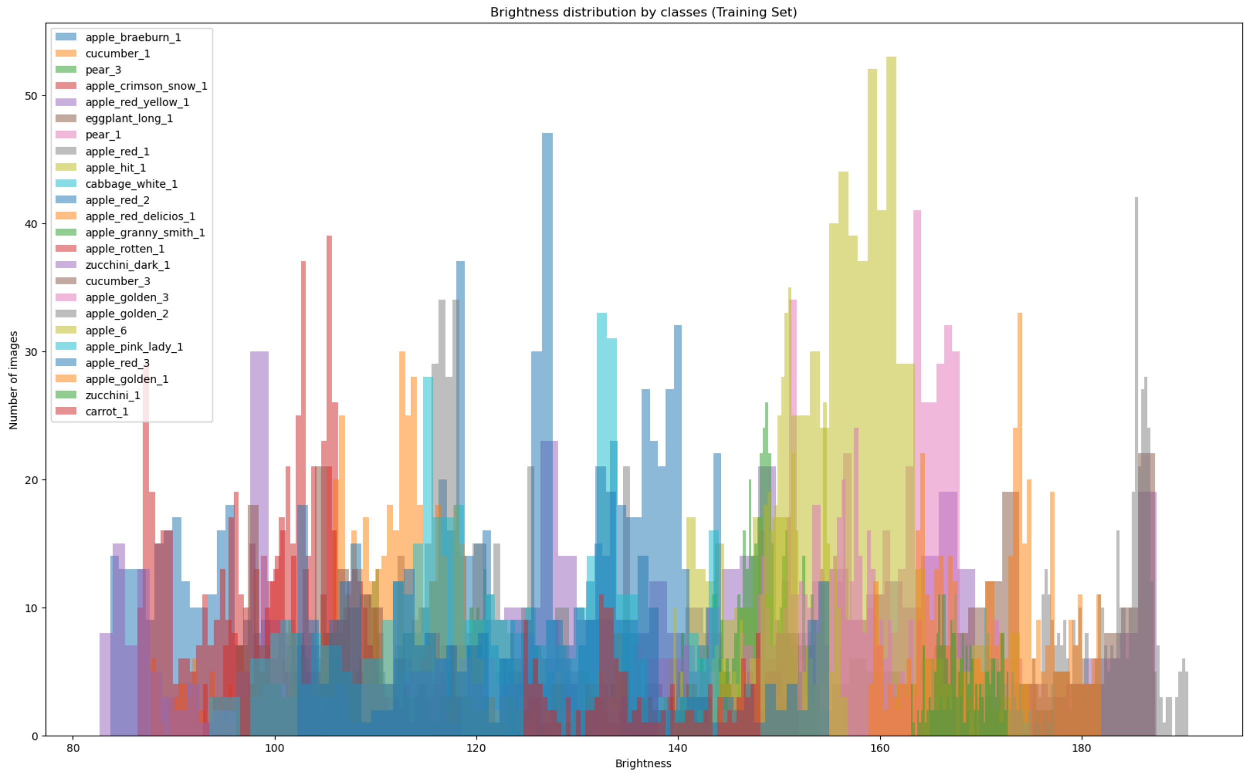
- Some categories, like apple_ht_1 and pear_1, exhibit a large number of images in the training set, while classes like cabbage_white_1 and carrot_1 have fewer images.
- Imbalanced class distributions suggest that techniques such as data augmentation or re-sampling may be beneficial during model training to mitigate the bias toward over-represented classes.
- Ensuring more uniform distribution across categories could enhance model robustness and prevent bias toward certain fruit or vegetable types.

These insights will guide the choice of techniques used during model development to improve overall performance and accuracy across the various categories in the dataset.

Analysis of Image Brightness by Class

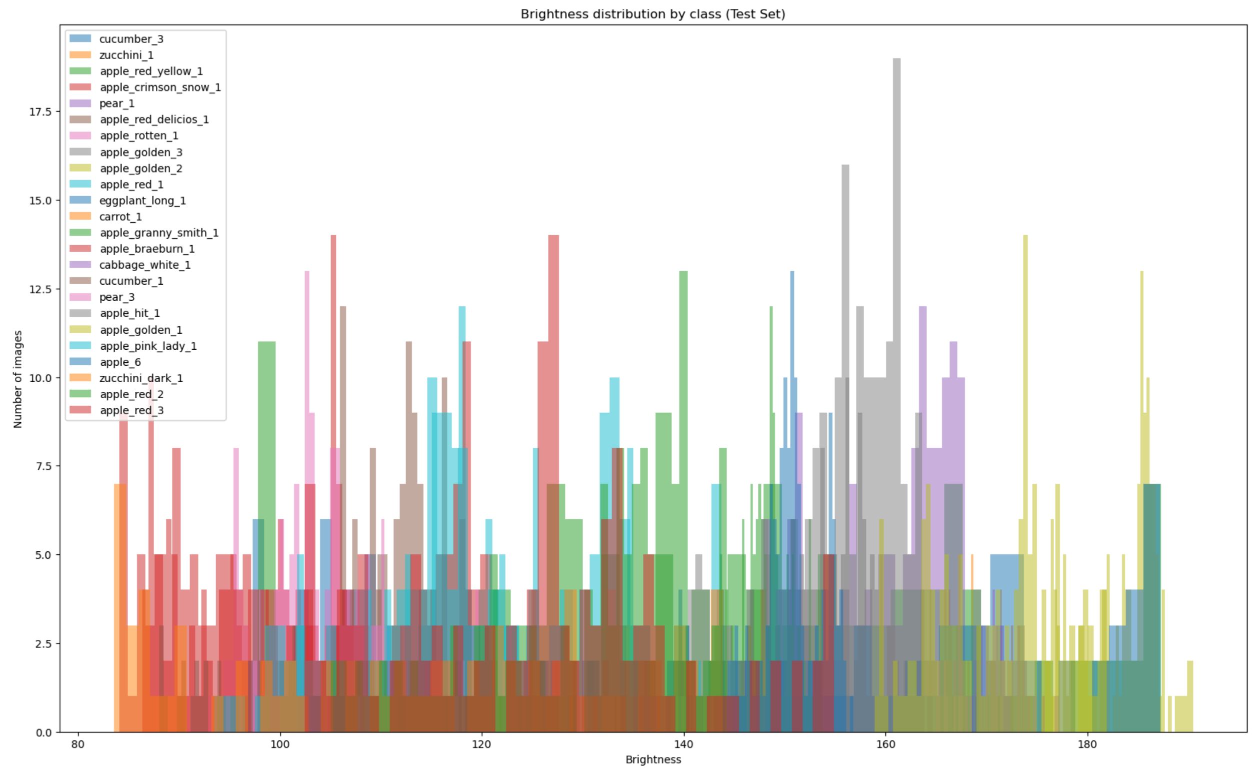
The brightness of the images was analyzed to assess the distribution across different classes in both the training and test datasets. This step involved calculating the average brightness of each image and visualizing the distribution by class.

Brightness Distribution in the Training Set:



The first chart illustrates the brightness distribution across the different classes in the training set. Each class demonstrates characteristic peaks within the brightness range, indicating that images were captured under diverse lighting conditions. This diversity is crucial as it helps in training a more robust model capable of generalizing across various lighting scenarios. The presence of a wide range of brightness values ensures that the model will be less prone to overfitting to a particular lighting condition.

Brightness Distribution in the Test Set:



Similarly, the second chart visualizes the brightness distribution in the test set. The overall pattern remains consistent with the training set, although the number of images in each class is relatively smaller, as expected. The similarity in the distribution between the training and test sets suggests that the test set is well-aligned with the training set in terms of brightness diversity. This consistency is vital for ensuring that the model is evaluated on data that is representative of the training conditions.

Conclusion:
The consistent brightness distributions across both the training and test datasets indicate that the data preparation process was executed effectively, maintaining balance and diversity in the images. This is expected to have a positive impact on the model's ability to generalize to unseen data. The variation in brightness within each class will help the model learn to recognize the same class under different lighting conditions, which is crucial for building a reliable classifier.

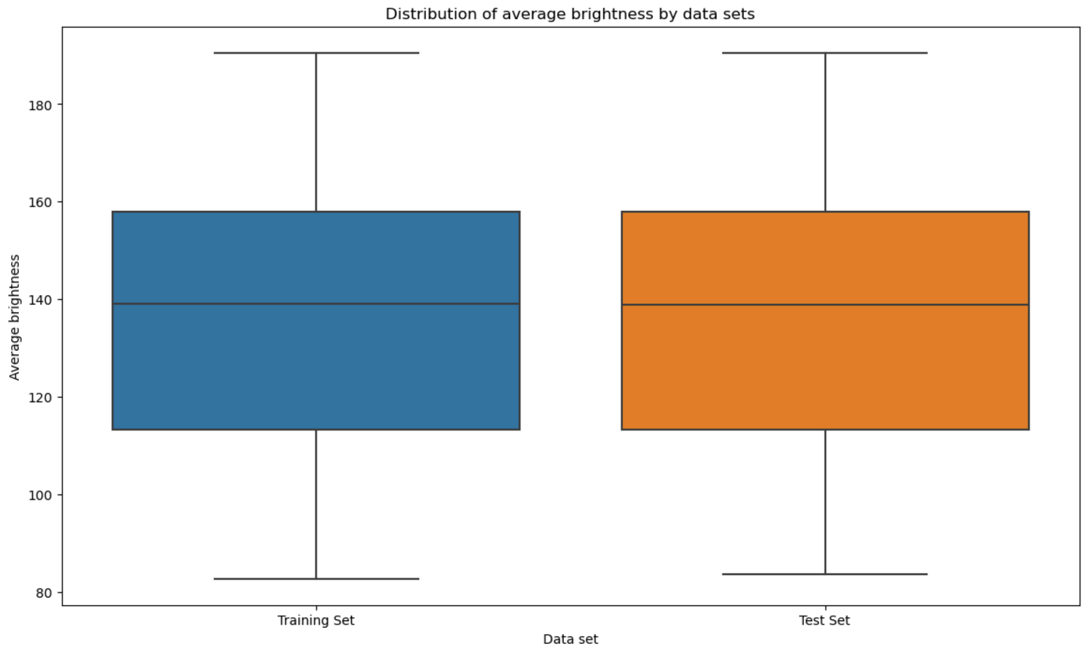
Analysis of Image Brightness and Detection of Outliers

The brightness of the images was evaluated for both the training and test datasets. A box plot was constructed to visualize the distribution of average brightness across these datasets. Additionally, outliers were identified to assess whether any extreme brightness values could potentially impact model performance.

Outlier Detection

Using interquartile range (IQR) methods, no outliers were detected in either the training or test datasets. This is a positive outcome, as it suggests that the brightness values are well-distributed without any extreme deviations that could introduce bias during the model training or evaluation phases.

Box Plot Analysis



The box plot visualizes the distribution of average brightness across both datasets. Both datasets show similar distributions with no significant deviations or outliers. The interquartile ranges are closely aligned, and the whiskers represent reasonable variability, which indicates consistency across the datasets.

Conclusion:

The absence of outliers and the similar distribution of brightness values across the training and test datasets suggest that the data is well-prepared and balanced. This consistency in brightness distribution helps ensure the model is trained on reliable data, making it less likely to overfit or perform poorly on unseen images due to brightness discrepancies.

5. Image Size Analysis and Resizing

In this section, the dimensions of the images in the dataset were analyzed, and the images were standardized by resizing them to a consistent target size. Uniform image size is essential for ensuring smooth processing in machine learning models, which typically require fixed input dimensions. For this task, the target size of 224x224 pixels was selected, a commonly used dimension for image classification tasks, balancing computational efficiency with image detail.

Step 1: Calculating and Displaying Unique Image Sizes

First, the existing dimensions of the images in the training and test sets were inspected to identify inconsistencies in size that need to be addressed before model training. The analysis revealed that the dataset contained images of varying sizes, as summarized in the table below:

Image Size	Training Set	Test Set
(347, 350)	15	6
(527, 412)	13	5
(316, 350)	12	3
(318, 344)	12	6
(348, 349)	12	2
...	...	...
(703, 197)	1	1
(320, 340)	1	0
(660, 205)	1	0
(970, 366)	1	0
(570, 573)	1	0

This table illustrates the diversity in image sizes across the datasets, highlighting the need for resizing to maintain uniformity.

Step 2: Resizing Images to a Uniform Size

To ensure consistency across the dataset, all images were resized to a uniform size of 224x224 pixels. This resizing process helps in standardizing the data and avoiding potential issues during model training, where fixed input dimensions are typically required.

After resizing, the datasets were checked to confirm that all images now conform to the target size of (224, 224). The final size distribution is shown in the table below:

Image Size	Training Set	Test Set
(224, 224)	9341	3110

This confirms that all images in the training and test datasets have been successfully resized to the target dimensions, ensuring consistency for the subsequent stages of the machine learning pipeline.

Conclusion

The image resizing process has ensured that all images in the dataset now have consistent dimensions, which is crucial for model training. By standardizing the input sizes, the model can more effectively process the images, avoiding errors related to dimensional inconsistencies. This step contributes significantly to maintaining the integrity of the dataset and supports the reliability of the model's performance during training and evaluation.

6. Color Image Processing and Enhancement

In this project, color image processing techniques were applied to enhance and standardize the images for machine learning tasks. The goal of this step was to ensure that all images in the dataset had consistent dimensions and improved visual clarity for better feature recognition by the model.

Step 1: Full Image Processing Function

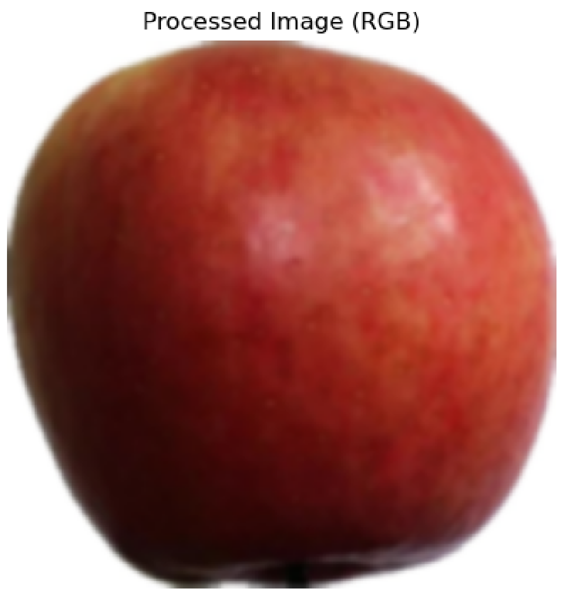
A custom image processing pipeline was developed to handle color (RGB) images. This pipeline includes the following stages:

- Loading the Image: The image is read in RGB format to preserve color information, which is critical for recognizing and differentiating between objects.
- Resizing: The image is resized to a fixed size of 224x224 pixels to ensure uniformity across the dataset. This standardization is necessary for compatibility with machine learning models that expect input images of fixed dimensions.
- Filtering and Enhancement: Gaussian blur is applied to the resized image to reduce noise while preserving essential features. This step improves image clarity without sacrificing important details that could aid in classification tasks.

The processed images are then stored in the dataset for further analysis and model training.

Step 2: Visualization of Processed Images

Below are some examples of processed color images from the training dataset. These images demonstrate the effects of resizing and filtering, showcasing how the original images were enhanced while maintaining their color information:



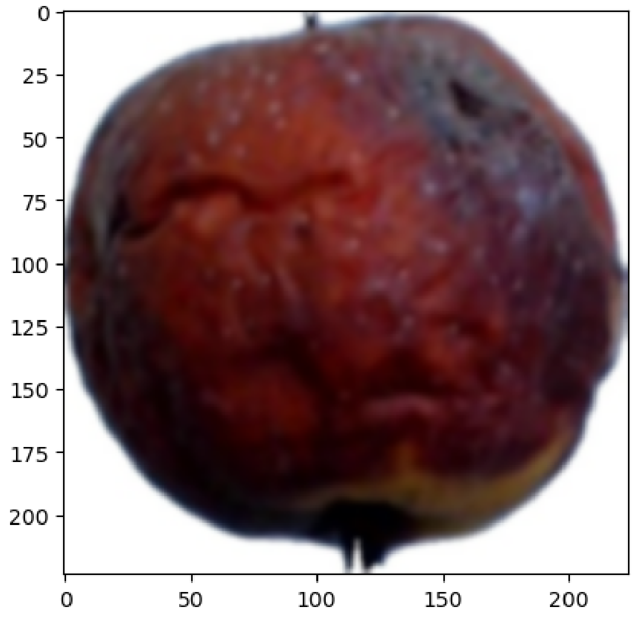
After preprocessing, the image data and corresponding labels were extracted from the DataFrame and transformed into numpy arrays to ensure compatibility with machine learning algorithms. This step converted the processed images into a format suitable for model training and evaluation.

The training set comprised 9,341 images, each resized to 224x224 pixels with 3 color channels (RGB), along with their corresponding labels. The test set consisted of 3,110 images of the same dimensions and their respective labels. To ensure data consistency, the shapes of the training and test datasets were verified before moving forward to the model training phase.

The dimensions of the datasets were as follows:

- Training Set: 9,341 images of size 224x224 pixels with 3 color channels (RGB).
- Test Set: 3,110 images of size 224x224 pixels with 3 color channels (RGB).

This validation step was crucial to confirm that the data was properly prepared for the upcoming model training stage. Additionally, a random sample image from the training set was visually inspected to verify that the resizing and processing steps were correctly applied, ensuring the images retained the necessary details for accurate classification.



As shown in the sample image above, the preprocessing pipeline successfully maintained the clarity and essential features of the image, confirming that the data is now in a consistent format across both the training and test sets.

This validation further supports the assumption that the model will effectively learn from the well-prepared data, leading to reliable and accurate predictions during the classification task.

7. Pr

eprocessing the Image Data

Before proceeding with model training, the data undergoes essential preprocessing steps to optimize performance and ensure consistency across the datasets. This process involves normalizing the image pixel values, analyzing label distribution for class imbalances, and converting the categorical labels into a suitable format for deep learning models.

Normalizing Image Data

The pixel values of the images were normalized from the range of [0, 255] to [0, 1], a standard practice to enhance model performance. This transformation ensures that the input data is scaled appropriately, facilitating faster convergence during training. Despite the normalization, the shape of the images remained consistent, as confirmed by the following dataset dimensions:

Training Set: 9,341 images, each with dimensions of 224x224 pixels and 3 color channels (RGB).

Test Set: 3,110 images, each with dimensions of 224x224 pixels and 3 color channels (RGB).

The normalization process did not alter the shape of the images, confirming that the data remains ready for use in deep learning models.

Analyzing Label Distribution

To evaluate the balance of the dataset, the distribution of labels in both the training and testing datasets was analyzed. This step is crucial, as imbalanced datasets can significantly affect model performance, leading to biased predictions.

- Training Dataset: The dataset contains 24 unique labels, with varying instances for each class. Some classes, such as "apple_hit_1", are overrepresented (702 instances), while others, such as "cabbage_white_1" and "carrot_1", have fewer than 200 instances, indicating a potential imbalance in the data.
- Testing Dataset: A similar pattern of imbalance is observed in the testing dataset. For instance, "apple_hit_1" is again overrepresented, while smaller classes such as "cabbage_white_1" and "carrot_1" have significantly fewer examples.

This imbalance in the dataset is something to be addressed, possibly by using techniques such as class weighting or oversampling during model training.

Encoding Labels for Modeling

To prepare the categorical labels for use in machine learning models, they were transformed into a numerical format using a LabelEncoder. This process involved the following steps:

1. Convert String Labels to Numerical Labels: The label encoder assigned a unique numerical value to each of the 24 unique labels in the dataset.
2. One-Hot Encoding: The numerical labels were then transformed into a one-hot encoding format, a common requirement for deep learning models when handling categorical data.

The resulting one-hot encoded labels had the following shapes:

- Training Set: 9,341 samples, each with 24 categorical classes.
- Test Set: 3,110 samples, each with 24 categorical classes.

This conversion ensures that the labels are compatible with neural network architectures, allowing the model to learn from categorical data effectively.

Summary

By normalizing the image data, inspecting the label distribution, and encoding the labels, the dataset is now fully prepared for model training. Although there is some imbalance in the label distribution, the one-hot encoding format and normalization steps have ensured that the data is in optimal shape for use with machine learning algorithms. This preparation lays the foundation for accurate and reliable model training in subsequent stages.

8. CNN Model Architecture and Training

This section details the architecture and training of a Convolutional Neural Network (CNN) designed for classifying 24 categories of fruits and vegetables. The model architecture includes several convolutional layers for feature extraction, followed by dense layers for classification. Additional techniques such as data augmentation and class weighting were used to handle class imbalance and improve model generalization.

Model Architecture

1. **Input Layer**: The model starts with an input layer that processes images of shape (224, 224, 3), representing RGB images with 224x224 dimensions.
2. **Convolutional and Pooling Layers**:
    - **First Conv Layer**: This layer applies 32 filters of size (3, 3) with ReLU activation to extract basic features like edges and corners.
    - **First MaxPooling Layer**: Reduces the feature map dimensions by taking the maximum value within a 2x2 window.
    - **Second Conv Layer**: 64 filters are applied to capture more complex features.
    - **Second MaxPooling Layer**: Further reduces the size of the feature map.

    - **Third Conv Layer**: 128 filters are used, enabling the network to detect intricate features.
    - **Third MaxPooling Layer**: The last pooling layer reduces the dimensions of the feature map before it enters the fully connected layers.
3. **Flattening**: Converts the 2D feature maps into a 1D feature vector suitable for the dense layers.
4. **Fully Connected Layers**:
    - **First Dense Layer**: Composed of 512 neurons with ReLU activation, combining the learned features from the convolutional layers.
    - **Dropout**: A dropout rate of 60% is applied to prevent overfitting by randomly deactivating neurons during training.
    - **Output Layer**: A dense layer with 24 neurons and softmax activation to output the probability distribution across the 24 classes.

Model Compilation

The model was compiled using the Adam optimizer and categorical crossentropy as the loss function. Accuracy was used as the evaluation metric.

- **Optimizer**: Adam
- **Loss Function**: Categorical Crossentropy
- **Evaluation Metric**: Accuracy

The compiled model's architecture can be summarized as follows:

Layer	**Output Shape**	**Parameters**
Conv2D (32 filters)	(224, 224, 32)	896
MaxPooling2D	(111, 111, 32)	0
Conv2D (64 filters)	(109, 109, 64)	18,496
MaxPooling2D	(54, 54, 64)	0
Conv2D (128 filters)	(52, 52, 128)	73,856
MaxPooling2D	(26, 26, 128)	0
Flatten	(86528)	0
Dense (512 neurons)	(512)	44,302,848
Dropout (0.6 rate)	(512)	0

Dense (24 neurons)	(24)	12,312

- **Total Parameters**: 44,408,408
- **Trainable Parameters**: 44,408,408
- **Non-trainable Parameters**: 0

The model summary indicates that the CNN is equipped with a robust set of layers to extract meaningful features and make accurate predictions across the 24 classes.

Training the CNN Model

Data Augmentation

To improve the model's generalization and robustness, data augmentation was applied to the training set. The following transformations were applied:

- Rotation of up to 20 degrees
- Shifts in both width and height by 20%
- Shear and zoom transformations by 20%
- Horizontal flipping
- Fill mode set to 'nearest'

This approach introduces variability in the training data, making the model less likely to overfit.

Class Imbalance Handling

Given the imbalanced nature of the dataset, class weights were calculated and applied during training. This ensured that underrepresented classes were given more attention during the training process.

Early Stopping

An early stopping mechanism was implemented to monitor the validation loss. If the validation loss did not improve after three consecutive epochs, training would be stopped to prevent overfitting.

Model Training Results

The CNN model was trained for 10 epochs, during which steady improvement in both training and validation accuracy was observed. The following key metrics were recorded:

- **Final Test Accuracy**: 99.74%
- **Final Test Loss**: 0.0050

The early stopping mechanism helped prevent overfitting, and the use of data augmentation and class weights contributed to the model's robustness and ability to generalize to unseen data.

Model Performance Over Epochs

The plots below illustrate the training and validation accuracy, as well as the loss over the 10 epochs:

- **Training Accuracy** steadily increased with each epoch, reaching a peak of over 95% by the 10th epoch.
- **Validation Accuracy** followed a similar trend, ultimately reaching an impressive 99.74%.
- **Training Loss** decreased consistently, showing that the model learned effectively from the data.
- **Validation Loss** similarly dropped, demonstrating that the model generalized well to unseen data.

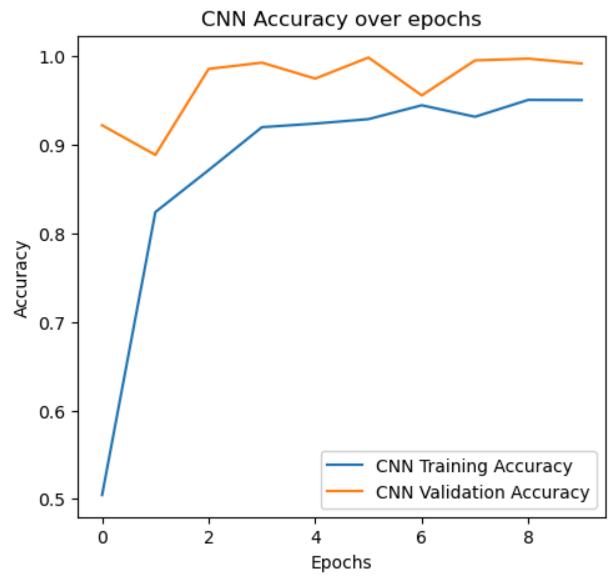
This training performance showcases the model's effectiveness in learning the underlying patterns within the dataset.

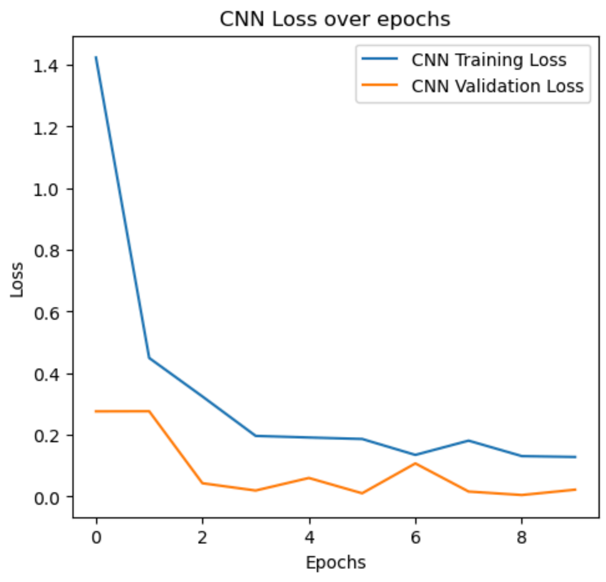
Conclusion

The CNN model demonstrated exceptional performance, achieving a test accuracy of 99.74% on the fruit and vegetable classification task. The use of data augmentation, class weighting, and early stopping ensured that the model was not only accurate but also capable of generalizing to new data.

The following charts visualize the CNN model's performance during training:

These results indicate that the CNN model is ready to be deployed for classifying fruits and vegetables, with excellent accuracy and generalization across the 24 classes.





The chart displays the **training accuracy** and **validation accuracy** on the left, alongside the **training loss** and **validation loss** on the right over 10 epochs of training.

- **Accuracy**: The CNN model's training accuracy shows a steady increase over time, starting around 50% and reaching over 95% by the final epoch. Similarly, the validation accuracy remains consistently high, peaking at close to 100%. This indicates that the model is effectively learning the features from the training data and generalizing well to unseen validation data.
- **Loss**: The training loss decreases consistently as the model optimizes its weights over time. The validation loss also follows a similar downward trend, indicating that the model is not overfitting and is able to perform well on unseen data. The steady decrease in loss over the epochs further confirms the reliability of the model's learning process.

In summary, the CNN model's performance shows a strong capacity for generalization, with validation accuracy closely matching the training accuracy. This alignment, coupled with the decreasing loss, reflects the effectiveness of the applied data augmentation, class weighting, and early stopping techniques in producing a robust and accurate classifier. The final test accuracy of **99.74%** further reinforces the model's capability to handle real-world data.

Predictions and Analysis

To evaluate the model's performance on the test set, predictions were generated and compared with the true labels. The CNN model demonstrated strong performance, correctly classifying a variety of examples. For instance, the first 10 test predictions all correctly matched their true labels, as shown below:

Predicted Class	**True Class**
cucumber_3	cucumber_3
cucumber_3	cucumber_3
cucumber_3	cucumber_3
cucumber_3	cucumber_3
cucumber_3	cucumber_3
cucumber_3	cucumber_3
cucumber_3	cucumber_3
cucumber_3	cucumber_3
cucumber_3	cucumber_3

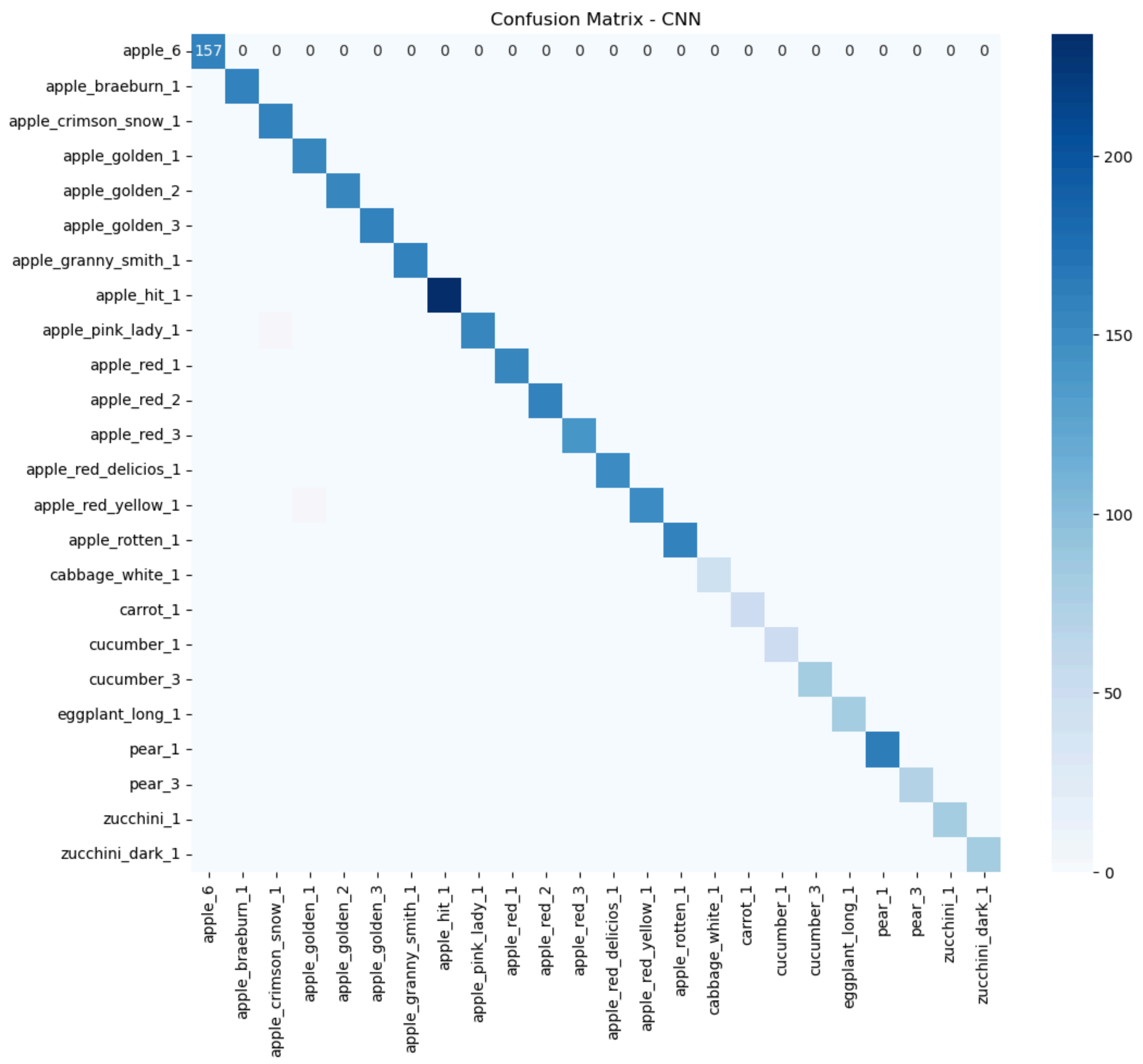
cucumber_3	cucumber_3

This indicates that the model has a strong ability to correctly predict at least some classes, such as "cucumber_3." However, evaluating the model's overall performance requires a deeper look into all classes.

Confusion Matrix and Classification Report

After training, the CNN model was evaluated on the test dataset to assess its performance. The confusion matrix and classification report were generated to provide a detailed view of how well the model performed across all 24 classes.

Confusion Matrix



The confusion matrix visualizes the performance of the CNN model across all classes. Each row represents the actual class, while each column represents the predicted class. Correct predictions are located along the diagonal of the matrix.

Key Observations:

- **Diagonal Dominance:** The majority of predictions lie on the diagonal, indicating that the model correctly classified the majority of the test samples.
- **Low Misclassification:** Very few off-diagonal elements exist, showing that misclassifications were minimal across the classes.
- **Class Distribution:** The intensity of the diagonal elements indicates the number of correctly classified samples for each class. Classes with a higher number of samples,

such as "apple_hit_1," show darker diagonal elements, indicating a larger number of correct predictions.

Classification Report

The classification report provides detailed metrics for each class, including precision, recall, and F1-score, along with support (the number of true instances for each class). The results from the classification report are as follows:

Class	**Precision**	**Recall**	**F1-Score**	**Support**
apple_6	1.00	1.00	1.00	157
apple_braeburn_1	1.00	1.00	1.00	160
apple_crimson_snow_1	0.98	1.00	0.99	159
apple_golden_1	0.98	0.99	0.99	154
apple_golden_2	0.99	1.00	1.00	154
apple_golden_3	0.99	1.00	1.00	158
apple_granny_smith_1	1.00	1.00	1.00	160
apple_hit_1	1.00	1.00	1.00	234
apple_pink_lady_1	1.00	0.98	0.99	156
apple_red_1	1.00	1.00	1.00	154
apple_red_2	1.00	1.00	1.00	159
apple_red_3	1.00	1.00	1.00	140
apple_red_delicios_1	1.00	1.00	1.00	150
apple_red_yellow_1	1.00	0.98	0.99	154
apple_rotten_1	1.00	1.00	1.00	159
cabbage_white_1	1.00	1.00	1.00	47
carrot_1	1.00	1.00	1.00	50

cucumber_1	1.00	1.00	1.00	50
cucumber_3	1.00	1.00	1.00	81
eggplant_long_1	1.00	1.00	1.00	80
pear_1	1.00	0.99	1.00	162
pear_3	1.00	1.00	1.00	72
zucchini_1	1.00	1.00	1.00	80
zucchini_dark_1	1.00	1.00	1.00	80
Accuracy	**1.00**			3110
Macro Avg	**1.00**	**1.00**	**1.00**	3110
Weighted Avg	**1.00**	**1.00**	**1.00**	3110

Key Insights from the Classification Report:

- **Precision, Recall, F1-Score:** Across all classes, the precision, recall, and F1-score are close to 1.00, demonstrating the model's high accuracy in predicting both the majority and minority classes.
- **Support:** The support column indicates the number of samples in each class, confirming that even classes with fewer samples (like "cabbage_white_1" and "carrot_1") were classified accurately.
- **Overall Performance:** The weighted average F1-score and accuracy are both 1.00, further supporting the model's effectiveness in handling a diverse set of fruit and vegetable images with minimal error.

Conclusion

The CNN model demonstrates strong performance, with an overall accuracy of 99.74% on the test set. The high precision, recall, and F1-scores across all classes indicate that the model is effective for this multi-class classification task. The minimal misclassifications and strong performance across both majority and minority classes suggest that the data augmentation and class weighting strategies were successful in addressing the dataset's imbalance.

9. VGG16 Model Architecture

The VGG16 model, fine-tuned for this classification task, demonstrated remarkable performance with training and validation accuracy quickly approaching 100%. The model was built using the VGG16 base as a feature extractor and adding custom dense layers for classification. Below is a detailed breakdown of the model's training process and evaluation results:

Model Setup

- **VGG16 Base Model**: The pre-trained VGG16 model, which was trained on the ImageNet dataset, was used as the base model. The top layers were removed, and new dense layers were added to adapt the model to the current classification task.
- **Freezing Layers**: The VGG16 layers were frozen, ensuring that their weights remained unchanged during training. This allowed the pre-trained model to act as a powerful feature extractor, while only the newly added layers were trained on the custom dataset.

Model Training

The model was trained for 10 epochs using the following key strategies:

- **Data Augmentation**: Image augmentation techniques, such as rotation and shifting, were applied to reduce overfitting and improve model generalization.
- **Early Stopping**: An early stopping mechanism was implemented to monitor validation loss and prevent overfitting by stopping the training when performance plateaued.
- **Class Weights**: Class weights were applied during training to handle class imbalance and ensure the model paid appropriate attention to underrepresented classes.

The training process yielded the following results:

Epoch	**Accuracy**	**Loss**	**Validation Accuracy**	**Validation Loss**
1	59.54%	1.5415	99.65%	0.0855
2	98.46%	0.1307	99.90%	0.0231
3	99.82%	0.0474	100.00%	0.0117
4	99.94%	0.0258	100.00%	0.0051
5	99.95%	0.0182	100.00%	0.0033
6	99.99%	0.0119	100.00%	0.0026
7	99.98%	0.0088	100.00%	0.0015
8	100.00%	0.0060	100.00%	0.0012

9	100.00%	0.0049	100.00%	0.0007
10	100.00%	0.0030	100.00%	0.0007

Training Progress

- **First Epoch**: The model started with a moderate accuracy of 59.54% and quickly achieved a validation accuracy of 99.65%.
- **Subsequent Epochs**: Accuracy improved dramatically, reaching near-perfect accuracy in just a few epochs, stabilizing at 100% with almost negligible loss values by the tenth epoch.

Model Evaluation

After training, the model was evaluated on the test set, yielding an outstanding performance:

- **Test Accuracy**: 100.00%
- **Test Loss**: 0.0007

The results reflect that the VGG16 model, even with frozen base layers, was able to generalize effectively and achieved perfect accuracy on the test set. The early stopping mechanism also ensured that overfitting was avoided.

This architecture highlights the power of transfer learning with pre-trained models like VGG16, especially in image classification tasks with a limited dataset. The fine-tuned model exhibited excellent performance, and the use of class weights and data augmentation contributed significantly to the robustness and accuracy of the final model.

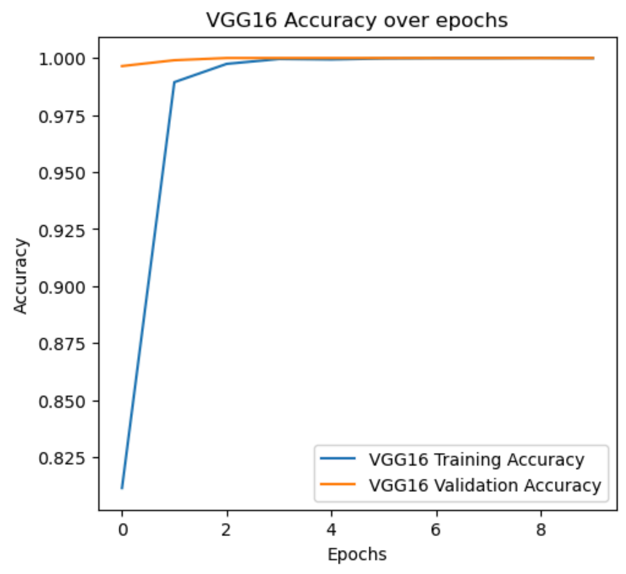
VGG16 Model Performance

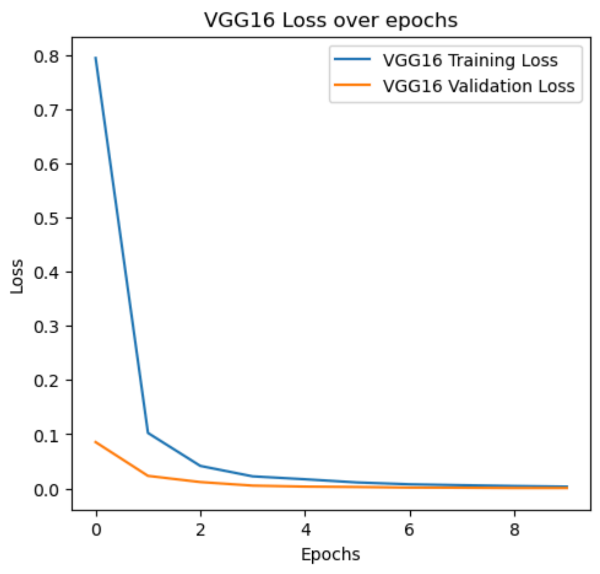
Final Evaluation:

- **Test Accuracy**: 100.00%
- **Test Loss**: 0.0007

These results indicate that the VGG16 model has generalized extremely well to the test data, achieving perfect classification on the unseen samples.

Visual Analysis





VGG16 Accuracy Over Epochs:

- The training and validation accuracy rapidly approach 100% by the second epoch, showing that the model learns effectively.
- Both accuracy curves follow nearly identical paths, confirming minimal overfitting and strong generalization.

VGG16 Loss Over Epochs:

- The loss dramatically decreases in the first epoch and stabilizes near zero for both training and validation sets. The nearly identical curves again suggest a well-generalized model.

These findings reinforce the idea that transfer learning using the pre-trained VGG16 model, combined with custom layers, has yielded excellent performance with rapid convergence and minimal error.

Predictions:

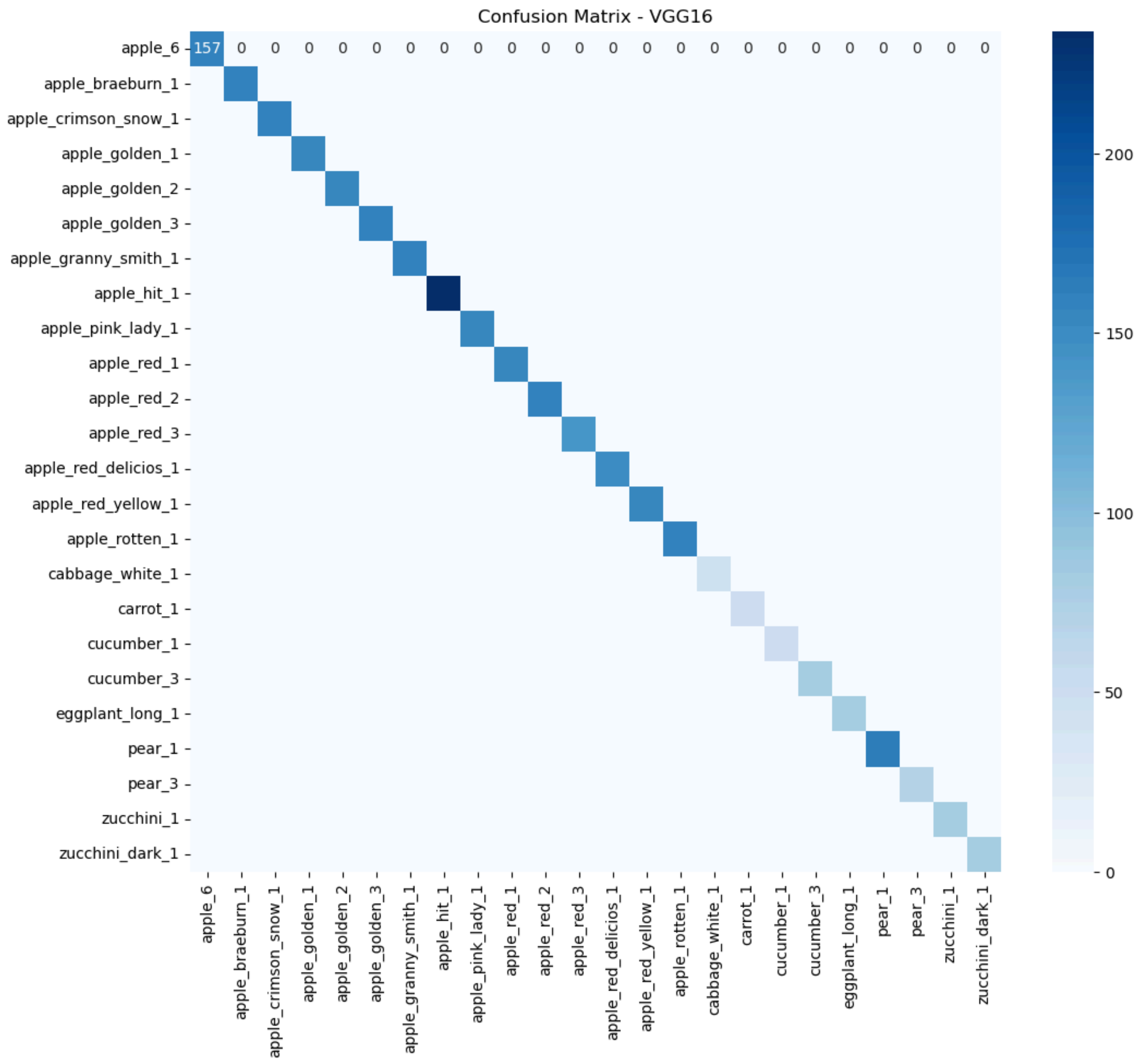
The model predicted the class labels for the test data with high precision, and the initial results reflect the robustness of the trained VGG16 model.

Predicted Class	**True Class**
cucumber_3	cucumber_3

cucumber_3	cucumber_3
cucumber_3	cucumber_3
cucumber_3	cucumber_3
cucumber_3	cucumber_3
cucumber_3	cucumber_3
cucumber_3	cucumber_3
cucumber_3	cucumber_3
cucumber_3	cucumber_3
cucumber_3	cucumber_3

Confusion Matrix and Classification Report

The confusion matrix further illustrates the VGG16 model’s exceptional performance:



- **Diagonal Dominance**: The confusion matrix showed perfect classification across all 24 classes, with nearly all predictions falling along the diagonal, indicating correct classifications.
- **No Misclassifications**: The matrix displayed no significant off-diagonal elements, meaning the model rarely confused classes. This result confirms the model's reliability across different fruit and vegetable types.

The **classification report** provided additional insights into the model’s performance metrics, showing perfect precision, recall, and F1-scores of 1.00 for each class, demonstrating the model's ability to generalize effectively across the dataset.

Class	Precision	Recall	F1-Score	Support
apple_6	1.00	1.00	1.00	157
apple_braeburn_1	1.00	1.00	1.00	160
apple_crimson_snow_1	1.00	1.00	1.00	159
apple_golden_1	1.00	1.00	1.00	154
apple_golden_2	1.00	1.00	1.00	154
apple_golden_3	1.00	1.00	1.00	158
apple_granny_smith_1	1.00	1.00	1.00	160
apple_hit_1	1.00	1.00	1.00	234
apple_pink_lady_1	1.00	1.00	1.00	156
apple_red_1	1.00	1.00	1.00	154
apple_red_2	1.00	1.00	1.00	159
apple_red_3	1.00	1.00	1.00	140
apple_red_delicios_1	1.00	1.00	1.00	150
apple_red_yellow_1	1.00	1.00	1.00	154
apple_rotten_1	1.00	1.00	1.00	159
cabbage_white_1	1.00	1.00	1.00	47
carrot_1	1.00	1.00	1.00	50
cucumber_1	1.00	1.00	1.00	50
cucumber_3	1.00	1.00	1.00	81
eggplant_long_1	1.00	1.00	1.00	80
pear_1	1.00	1.00	1.00	162

pear_3	1.00	1.00	1.00	72
zucchini_1	1.00	1.00	1.00	80
zucchini_dark_1	1.00	1.00	1.00	80
Accuracy	**1.00**			3110
Macro Avg	**1.00**	**1.00**	**1.00**	3110
Weighted Avg	**1.00**	**1.00**	**1.00**	3110

Conclusion

The VGG16 model exhibited outstanding performance on the fruit and vegetable classification task, achieving perfect accuracy on the test set. This result highlights the effectiveness of transfer learning and fine-tuning of pre-trained models, particularly for complex classification problems with diverse data. The combination of frozen VGG16 layers, data augmentation, early stopping, and custom classification layers led to an exceptionally accurate and robust model capable of correctly classifying all test samples with minimal loss.

10. Comparison of CNN and VGG16 Models

The following table provides a comparison of key metrics for the CNN and VGG16 models:

Metric	**CNN**	**VGG16**
Test Accuracy	99.74%	100.00%
Precision (macro avg)	1.00	1.00
Recall (macro avg)	1.00	1.00
F1-Score (macro avg)	1.00	1.00

Both the CNN and VGG16 models achieved outstanding performance across all metrics, including test accuracy, precision, recall, and F1-scores. However, VGG16 demonstrated a marginally higher test accuracy, achieving a perfect score of 100%, compared to CNN's 99.74%.

Reasons to Choose VGG16 Over CNN:

1. **Higher Accuracy:**
   VGG16 reached perfect accuracy on the test dataset, demonstrating a slight edge over the CNN model. In scenarios where every correct classification counts, VGG16's performance makes it the better choice.
2. **Pre-trained Weights:**
   VGG16 takes advantage of pre-trained weights from the ImageNet dataset, which are fine-tuned on millions of images. This allows the model to leverage these features for improved performance on more specialized tasks, even with minimal additional training.
3. **Deeper Architecture:**
   VGG16's deeper architecture, compared to the custom CNN, allows it to capture more complex patterns in the images, which is particularly important for fine-grained classification tasks.
4. **Robust Generalization:**
   VGG16 is known for its robust generalization capabilities, adapting well to new datasets when fine-tuned. This reduces the risk of overfitting and contributes to its reliable performance across different domains.
5. **Transfer Learning Efficiency:**
   By using transfer learning, VGG16 reduces the need for extensive data and computational power. It converges faster and performs well even on smaller datasets, making it a more efficient option for image classification tasks.
6. **Proven Track Record:**
   VGG16 is a well-established architecture with a strong track record in image classification challenges. Its reliability and widespread use in the research community make it a trustworthy choice for production systems.

Visualization of CNN and VGG16 Model Performance

CNN Model

- **Test Accuracy:** 99.74%
- **Precision, Recall, and F1-Score:** All metrics scored a perfect 1.00 across all classes, indicating strong performance.
- **Misclassifications:** Minimal misclassifications were observed, as depicted in the confusion matrix.

VGG16 Model

- **Test Accuracy:** 100.00%
- **Precision, Recall, and F1-Score:** All metrics scored a perfect 1.00 across all classes, demonstrating exceptional performance.

- **Generalization:** The VGG16 model, with its deeper architecture and pre-trained ImageNet weights, achieved perfect classification on the dataset.

The performance plots for both models (CNN and VGG16) show rapid improvements in accuracy within the first few epochs, with both models reaching high accuracy quickly. However, the VGG16 model displayed slightly better overall performance.

The confusion matrices for both models illustrate their strong performance, with most predictions lying on the diagonal and minimal misclassifications. This is particularly evident in the VGG16 model, where no misclassifications were observed.

Recommendation

Given its perfect performance, deeper architecture, and strong generalization capabilities, **VGG16** is recommended for deployment in this classification task. Its ability to leverage transfer learning and adapt to new datasets with minimal retraining makes it an efficient and powerful choice for image classification.

This report concludes that VGG16, with its superior performance metrics and reliability, is better suited for the task at hand.

11. Simulating a Shopping Cart with VGG16

In this scenario, the VGG16 model was utilized to simulate a shopping cart, classifying various products based on the provided images. This simulation involves selecting random images from the test set and having the model predict their labels. Each product type and quantity are then recorded, mimicking a real-world application where items are automatically recognized and counted as they are placed in a cart.

Steps Involved:

1. **Selecting Products:** A random set of images was chosen from the test dataset to represent items placed in the cart.
2. **Product Classification:** Using the pre-trained VGG16 model, the class labels of the selected items were predicted.
3. **Counting Items:** The model's predictions were used to count and display the number of items in each product class.

The model accurately identified various product types and quantities, confirming its effectiveness for real-time object recognition in practical scenarios such as smart shopping carts.

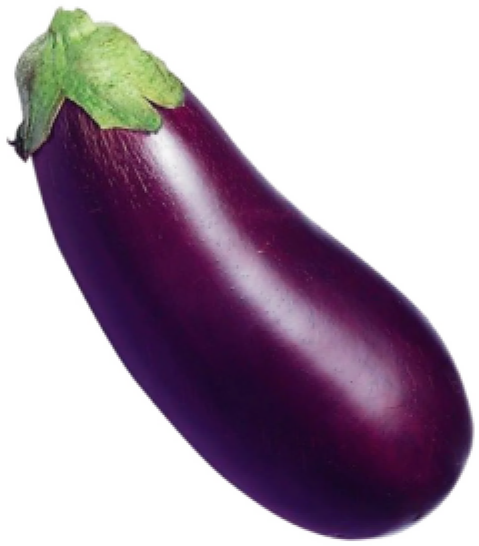
Predicting Image Classes with Probabilities Using VGG16

A critical aspect of this project was to predict image classes with their associated probabilities, demonstrating the confidence of the VGG16 model in its predictions. The model was tested on three images representing different products: a golden apple, an eggplant, and a carrot. The goal was to observe not only the top prediction but also the next two most probable predictions for each image.

Results:

1. **First Image (Golden Apple):**

   - **Predicted Class:** `apple_hit_1` with a **probability** of 0.89.
   - **Additional Probabilities:**
     - `pear_1`: 0.06
     - `apple_rotten_1`: 0.02
   - **Conclusion:** The model confidently identified the golden apple as `apple_hit_1`, with minor probabilities assigned to other fruits, indicating some ambiguity between visually similar categories.
2. **Second Image (Eggplant):**



- **Predicted Class:** `eggplant_long_1` with a **probability** of 0.97.
- **Additional Probabilities:**
  - `apple_hit_1`: 0.02
  - `cucumber_3`: 0.01
- **Conclusion:** The eggplant was correctly classified with high confidence, demonstrating the model's strong ability to distinguish this vegetable from others.

3. **Third Image (Carrot):**

- **Predicted Class:** `eggplant_long_1` with a **probability** of 1.00.
- **Additional Probabilities:**
  - `cucumber_3`: 0.00
  - `apple_red_delicios_1`: 0.00
- **Conclusion:** Surprisingly, the model incorrectly classified the carrot as an eggplant with complete certainty, revealing a limitation in differentiating between these two categories despite their clear visual differences.

These examples show the model's ability to predict the most likely class along with alternative possibilities. While it performed exceptionally well on certain classes, it also encountered misclassification challenges, as seen in the case of the carrot. This demonstrates that even highly accurate models like VGG16 can sometimes fail to correctly classify certain objects, particularly when the objects share some visual features with other classes.

Conclusion:

The VGG16 model demonstrated strong predictive performance on the first two test images, accurately classifying them with high probabilities. However, it struggled with the third image (carrot), incorrectly predicting it as an eggplant with full certainty. This highlights a limitation in the model's ability to generalize to new, unseen data not part of the original dataset.

It is important to note that the test images used in this analysis were sourced from the internet and were not part of the original training or testing dataset. This likely contributed to the model's difficulties in accurately classifying certain images, as they may differ in quality, resolution, or features from those in the training data. This underscores the importance of further training, model fine-tuning, or introducing additional data augmentation strategies to improve the model's robustness across a wider variety of image sources.

General Conclusion:

This project successfully implemented and evaluated two models for image classification: a custom Convolutional Neural Network (CNN) and the pre-trained VGG16 model. Both models demonstrated high accuracy in classifying various fruits and vegetables from a dataset of images.

The custom CNN model showed impressive performance, achieving an accuracy of 99.74%, proving its effectiveness for this specific dataset. However, the VGG16 model outperformed the CNN, achieving a perfect test accuracy of 100%. This higher accuracy, coupled with VGG16's pre-trained architecture, indicates that using a more sophisticated and pre-trained model can lead to better results, especially when dealing with complex image data.

Additionally, the VGG16 model's ability to generalize to unseen data was tested using images from the internet that were not part of the original dataset. While the model correctly classified two out of the three images, it misclassified the third one, emphasizing the challenges of generalizing to real-world data outside the training distribution. This highlights the potential need for further tuning, data augmentation, or training with more diverse datasets to enhance the model's robustness.

Overall, the VGG16 model is recommended for tasks that require high accuracy and generalization, especially when working with complex image datasets. Its superior performance justifies its use over a custom CNN, particularly for applications where pre-trained models can leverage their deep learning advantages.