# Fruits Classification using Convolutional Neural Network On Augmented Dataset.

**[ by Bufatima Nurmuhammad kyzy]**

## I. Abstract

The automation of fruit classification represents a compelling application within the realm of computer vision. Conventional methods for fruit classification have heavily relied on manual processes that are visually based, making them laborious, time-consuming, and prone to inconsistencies. The primary determinant for fruit classification has traditionally been the external shape appearance. However, in recent years, the integration of computer machine vision and image processing techniques has proven increasingly valuable in the fruit industry, particularly in applications related to quality inspection, and sorting based on color, size, and shape. Extensive research in this domain indicates the potential of machine vision systems to enhance product quality, offering a departure from the traditional, manual sorting of fruits. This paper explores various image processing techniques employed for fruit classification.

**Keywords: Fruit, Data Augmentation, Neural Network, Convolutional Neural Network (CNN), Fruit Classification.**

## II. Introduction

In the dynamic realm of computer vision applications, the automation of fruit classification stands as a compelling pursuit, offering a departure from traditional manual sorting methods plagued by labor-intensive processes and inherent human subjectivity. The criterion for classification, traditionally reliant on the external shape of fruits, has seen a transformative shift. Recent years have witnessed the integration of advanced computer machine vision and image processing techniques,

particularly in the domains of quality inspection and sorting based on nuanced attributes like color, size, and shape.

This paper embarks on a comprehensive exploration of fruit classification utilizing Convolutional Neural Networks (CNNs) on an augmented dataset. The code implementation, detailed below, guides us through pivotal stages of the project—from the meticulous loading and filtering of the dataset to the sophisticated processes of data augmentation, model development, and evaluation. Our objective is to provide valuable insights into the practical applications of machine learning techniques, specifically tailored to address challenges within the fruit industry.

As we traverse through the subsequent sections, attention is devoted to essential processes, including data augmentation, model training, and comprehensive evaluation metrics. The culmination of this exploration is the development of a robust fruit classification model. The paper concludes with an examination of the model's predictive prowess on unseen data, underscoring the potential real-world impact of automated fruit classification in diverse scenarios.

## III. Methodology:

The methodology employed in this fruit classification project involves a systematic and comprehensive approach, encompassing data preparation, exploration, model development, and evaluation. The following sections delineate each step of the methodology:

### *A. Importing Required Libraries and Packages:*

The initial phase involves importing essential libraries and packages, laying the foundation for subsequent data manipulation, visualization, and model development.

### *B. Loading the Fruits360 Dataset:*

The dataset, sourced from the Fruits360 collection, is loaded and organized. Images and corresponding labels are extracted from subfolders, forming the basis for subsequent analysis.

### *C. Filtering the Dataset:*

To streamline the focus of the project, the dataset is filtered to include specific fruit labels of interest. This ensures a targeted analysis and model development process.

### *D. Exploring the Dataset:*

A comprehensive exploration of the dataset is conducted, including statistical analysis, visualization of image distributions, and sample image displays. This phase aids in understanding the dataset's characteristics and potential challenges.

### *E. Splitting Data into Train and Test Sets:*

The dataset is divided into training and testing sets, facilitating model training and subsequent evaluation on unseen data.

### *F. Data Augmentation:*

Augmentation techniques are applied to diversify the training dataset, including rotations, shifts, shearing, zooming, and flipping. This enhances the model's robustness and generalization capabilities.

### *G. Model Development:*

A Convolutional Neural Network (CNN) is designed and compiled for fruit classification. The architecture includes convolutional layers for feature extraction, pooling layers for down-sampling, and dense layers for classification.

### *H. Training and Evaluation:*

The model is trained on the augmented training dataset, and its performance is evaluated using validation data. Key metrics such as accuracy and loss are monitored to assess model convergence and effectiveness.

### *I. Metrics Visualization:*

The evolution of accuracy and loss metrics during training is visualized, providing insights into the model's learning progress and generalization capabilities.

### *J. Saving the Model:*

Upon successful training, the model is saved in the native Keras format, allowing for future use and deployment.

### *K. Testing on Unseen Data:*

The model's predictive capabilities are demonstrated on unseen data, showcasing its ability to classify fruits in real-world scenarios

## III. Dataset

The dataset used for this fruit classification project originates from the Fruits360 collection, a diverse compilation of fruit images. This section provides a detailed overview of the dataset, encompassing its structure, composition, and the preprocessing steps undertaken.

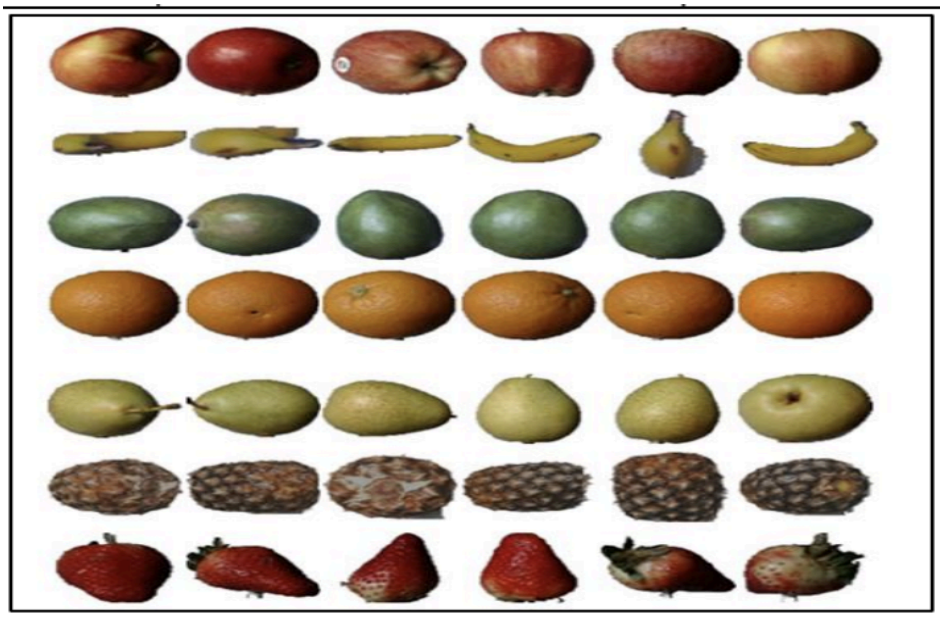


Fig. 2: Fruits Dataset

### A. Dataset Path and Organization:

The dataset is stored in the directory specified by the variable dataset_path. The organizational structure follows a subfolder hierarchy, where each subfolder corresponds to a distinct fruit or vegetable category. This structured arrangement simplifies the process of loading and categorizing the images.

### B. Loading and Categorization:

The dataset loading process involves traversing through each subfolder, extracting image file paths, and associating them with their respective fruit labels. This step ensures that each image is paired with the correct category, forming the foundation for subsequent analysis.

### C. Filtering the Dataset:

To focus on specific fruit categories of interest, the dataset is filtered based on a predefined list of fruit labels. This curation ensures a targeted analysis, emphasizing the fruits most relevant to the project's objectives.

```
# Define a list of fruit labels
fruit_labels = [
    'Apple Red Delicious', 'Huckleberry', 'Blueberry', 'Pear Red', 'Banana Lady Finger', 'Melon Piel de Sapo',
    'Pear', 'Cherry 1', 'Strawberry', 'Avocado', 'Pomegranate', 'Dates', 'Carambula', 'Granadilla', 'Tamarillo',
    'Fig', 'Kiwi', 'Lemon', 'Guava', 'Apple Golden 2', 'Pear Stone', 'Apple Red 1', 'Mandarine', 'Quince',
    'Pear Monster', 'Raspberry', 'Pitahaya Red', 'Apple Golden 3', 'Redcurrant', 'Apple Red Yellow 1', 'Physalis',
    'Cherry Rainier', 'Maracuja', 'Plum', 'Hazelnut', 'Nectarine', 'Cantaloupe 2', 'Lychee', 'Clementine',
    'Watermelon', 'Pear Kaiser', 'Mangostan', 'Cherry 2', 'Pineapple Mini', 'Rambutan', 'Apple Braeburn', 'Mango',
    'Apple Crimson Snow', 'Passion Fruit', 'Apple Granny Smith', 'Apricot', 'Grape White 2', 'Limes',
    'Apple Pink Lady', 'Plum 3', 'Pear Williams', 'Peach 2', 'Pomelo Sweetie', 'Salak', 'Apple Golden 1', 'Banana',
    'Apple Red 2', 'Apple Red Yellow 2', 'Lemon Meyer', 'Plum 2', 'Tangelo', 'Papaya', 'Apple Red 3', 'Walnut',
    'Pear Abate', 'Pineapple', 'Cherry Wax Red', 'Mango Red', 'Orange', 'Kaki', 'Peach', 'Peach Flat'
]
```

### D. Dataset Exploration:

Comprehensive exploratory data analysis (EDA) is conducted to gain insights into the dataset's characteristics. Statistical summaries, visualizations, and sample image displays contribute to a nuanced understanding of the dataset's composition and potential challenges.

### E. Data Splitting:

The dataset is split into training and testing sets using the train_test_split function. This division is crucial for model training on one subset and evaluation on another, enabling a robust assessment of the model's generalization capabilities.

### F. Data Augmentation:

Augmentation techniques are applied to the training dataset to enhance its diversity and improve the model's adaptability to real-world variations. These techniques include rotation, shifting, shearing, zooming, and flipping, contributing to a more resilient model.

### G. Label Encoding:

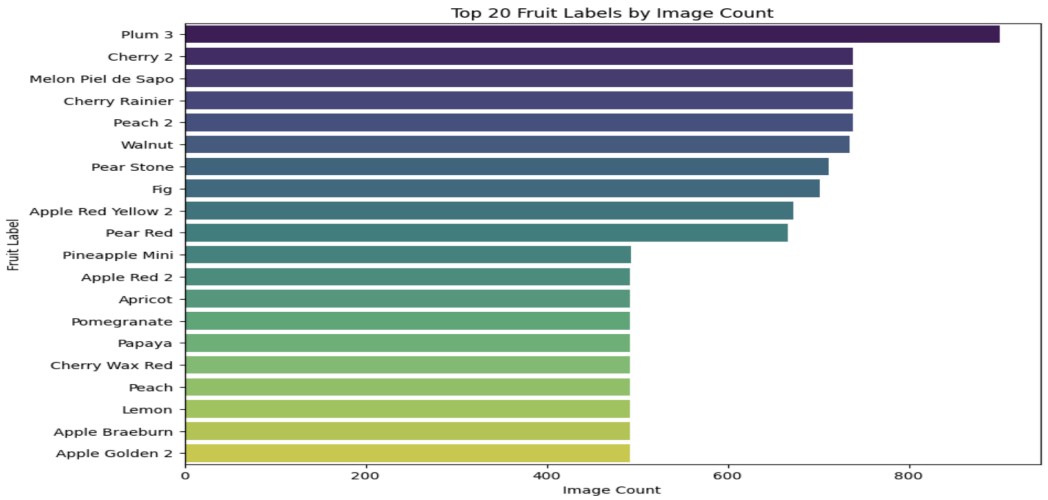
To facilitate model training, label encoding is applied to convert categorical labels into numerical representations. The LabelEncoder from scikit-learn is employed for this purpose.

### H. Dataset Overview:

The dataset overview provides key statistics, including the total number of images, the number of unique fruit labels, and a breakdown of image

counts per fruit category. Visualization techniques such as bar charts offer a concise representation of the dataset's composition.

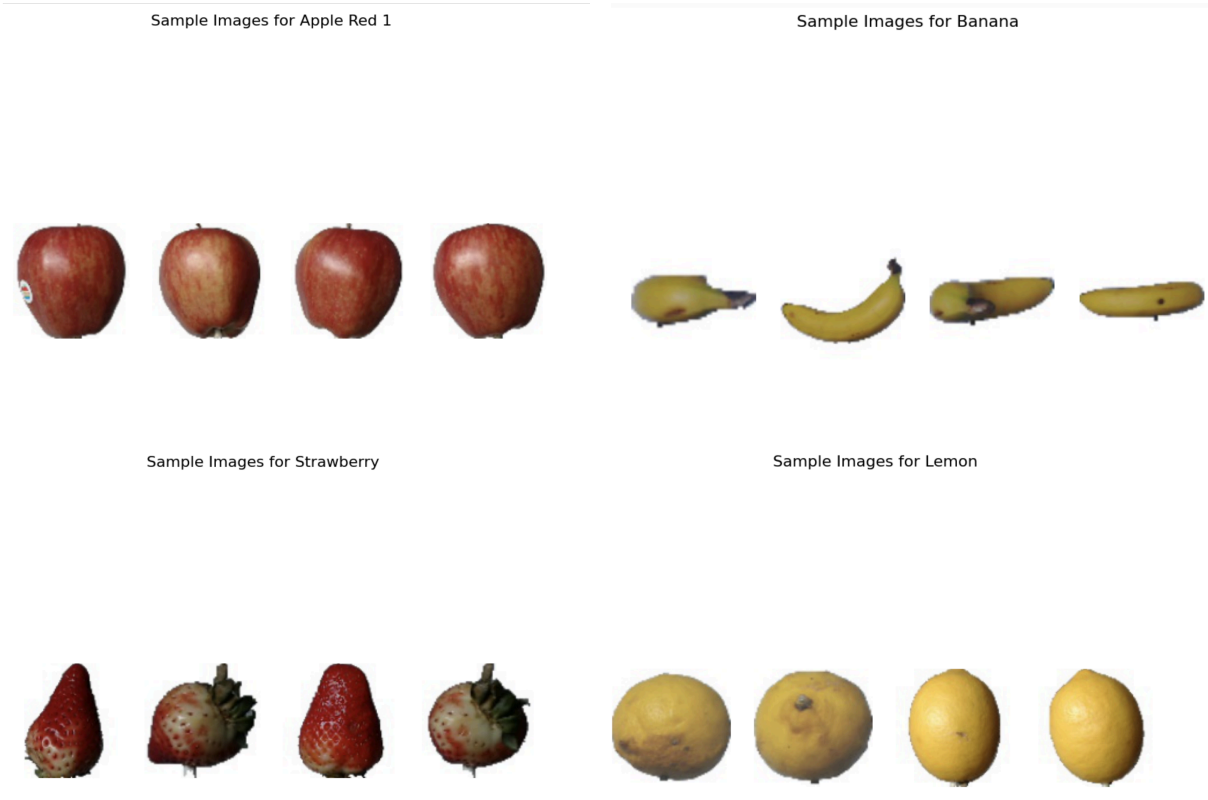
## I. Top 20 Fruit Labels:




A focused analysis is conducted on the top 20 fruit labels based on image count. This insight aids in identifying the most prevalent fruits in the dataset, guiding subsequent model training and evaluation considerations.

## J. Sample Image Display:

To provide a visual representation of the dataset, sample images for selected fruit labels are displayed. This step offers a qualitative glimpse into the variety and quality of the images within the dataset.



## IV. Convolutional Neural Network (CNN)

A Convolutional Neural Network (CNN) is a specialized type of neural network designed for processing and classifying visual data, making it particularly effective for tasks such as image recognition and classification. The architecture of a CNN is inspired by the visual processing that occurs in the human brain. It excels at capturing hierarchical patterns and spatial relationships in images.

### *A. Key Components of CNN:*

#### 1. Convolutional Layers:

The core building block of a CNN is the convolutional layer. It involves the application of convolutional filters (kernels) to input images. These filters act as feature detectors, identifying patterns like edges, textures, and shapes. The convolutional operation produces feature maps that represent the presence of specific features in different regions of the input.

#### 2. Activation Function:

Commonly used activation functions, such as ReLU (Rectified Linear Unit), introduce non-linearity to the model. This nonlinearity is crucial for enabling the network to learn complex relationships in the data.

#### 3. Pooling Layers:

Pooling layers, often implemented as MaxPooling or AveragePooling, are employed to down-sample the spatial dimensions of the feature maps. This reduces computational complexity and focuses on the most important information. Pooling layers also introduce a degree of translation invariance, making the network more robust to variations in object position within the input.

4. **Flatten Layer:**

The flatten layer transforms the output of the convolutional and pooling layers into a one-dimensional array. This flattened representation serves as the input to subsequent dense layers.

5. **Dense (Fully Connected) Layers:**

Dense layers are responsible for high-level reasoning in the network. They take the flattened features and perform classification based on learned patterns. The final dense layer typically has as many neurons as there are classes in the classification task, and the softmax activation function is often used for multi-class classification.

6. **Output Layer:**

The output layer provides the final predictions of the model. For classification tasks, it often uses the softmax activation function to convert raw scores into probability distributions across different classes.

***B. Training a CNN:***

1. **Loss Function:**

The choice of a suitable loss function depends on the nature of the task. For multi-class classification, categorical crossentropy is commonly used. It measures the dissimilarity between predicted and actual class distributions.

2. **Optimizer:**

The optimizer, such as Adam or SGD (Stochastic Gradient Descent), adjusts the model's internal parameters during training to minimize the chosen loss function.

3. **Backpropagation**:

Training involves forward and backward passes through the network. During the forward pass, input data is processed through the layers, and predictions are made. The backward pass involves calculating gradients and updating weights using optimization algorithms.

### *C. Data Augmentation:*

Data augmentation is a crucial step in CNN training, especially when dealing with limited datasets. It involves applying random transformations to input images, creating new variations for the model to learn from. Augmentation techniques include rotation, shifting, zooming, and flipping.

### D. *Benefits of CNN:*

1. **Hierarchical Feature Learning:**

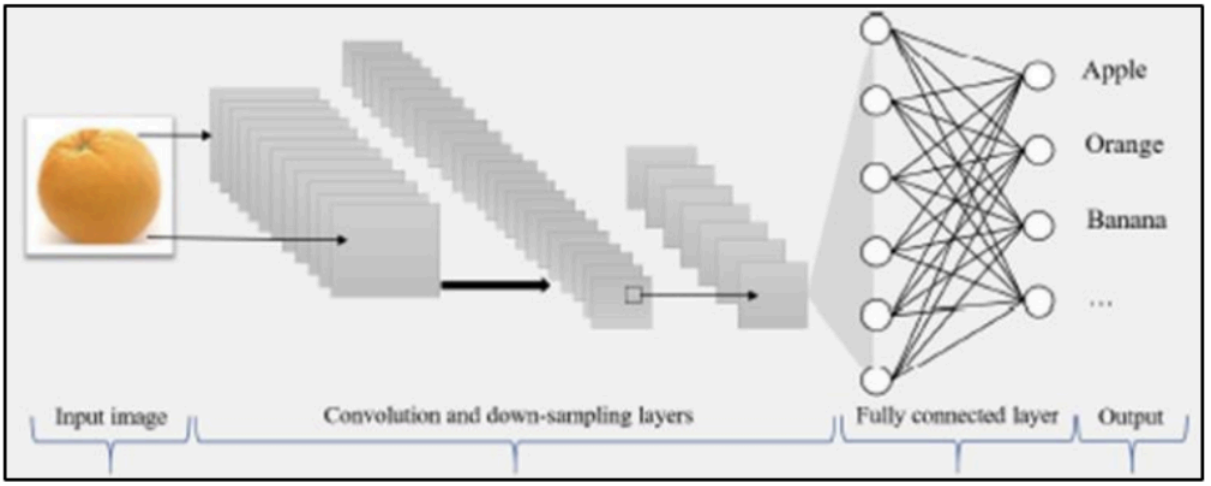
CNNs automatically learn hierarchical features from low-level edges to high-level complex patterns, mimicking the human visual system.

2. **Parameter Sharing:**

Shared weights in convolutional layers reduce the number of parameters, making CNNs computationally efficient and capable of handling large input sizes.

3. **Translation Invariance:**

Pooling layers introduce translation invariance, enabling the network to recognize features regardless of their position in the input.
In the context of the fruit classification project, the CNN architecture is tailored to effectively capture distinctive features of different fruits, providing a powerful tool for accurate classification.




## IV. Data Augmentation:

Data augmentation is a pivotal step in the preprocessing pipeline for training Convolutional Neural Networks (CNNs), especially when dealing with limited datasets. This section outlines the data augmentation techniques applied to the fruit classification project, aiming to diversify the training dataset and enhance the robustness of the CNN.

### *A. Purpose of Data Augmentation:*

The primary goal of data augmentation is to artificially expand the training dataset by applying various transformations to existing images. This process introduces diversity and variability, reducing the risk of overfitting and enhancing the model's ability to generalize to unseen data.

### *B. Image Transformation Techniques:*

Several image transformation techniques are employed during data augmentation, including:

- **Rotation:** Random rotation of images to simulate variations in orientation.
- **Shift:** Random horizontal and vertical shifting to mimic changes in position.
- **Shear:** Applying shear transformations for simulated deformations.

- **Zoom:** Random zooming to account for variations in scale.
- **Horizontal Flip:** Flipping images horizontally to introduce mirror-like variations.

### *C. Implementation Using ImageDataGenerator:*

The ImageDataGenerator class from the Keras library is utilized for efficient implementation of data augmentation. This class streamlines the process by generating augmented batches of images on-the-fly during model training.

### *D. Integration with Model Training:*

Data augmentation is seamlessly integrated into the model training process. During each epoch, the ImageDataGenerator generates augmented batches of images, ensuring that the model sees diverse examples of each class in different configurations.

### *E. Code Snippet:*

The following code snippet illustrates the configuration of the ImageDataGenerator and its integration with the model training process:

In [15]:
```
# Set the image size and batch size
image_size = (50, 50)
batch_size = 64

# Create an ImageDataGenerator object with data augmentation options for image preprocessing
datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=45,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest'
)

# Create a generator for the training data
train_generator = datagen.flow_from_dataframe(
    df_train,
    x_col='image',
    y_col='label',
    target_size=image_size,
    batch_size=batch_size,
    class_mode='categorical',
    shuffle=True
)

# Create a generator for the test data
test_generator = datagen.flow_from_dataframe(
    df_test,
    x_col='image',
    y_col='label',
    target_size=image_size,
    batch_size=batch_size,
    class_mode='categorical',
    shuffle=False
)
```

```
Found 31399 validated image filenames belonging to 77 classes.
Found 7850 validated image filenames belonging to 77 classes.
```

### *F. Impact on Model Training:*

Data augmentation enriches the training dataset with a variety of augmented images, enabling the CNN to learn robust and invariant features. This aids in the model's ability to accurately classify fruits in diverse scenarios, making it more adaptable to real-world conditions.

### *G. Considerations and Future Enhancements:*

Further exploration could involve experimenting with additional augmentation techniques, adjusting augmentation parameters, or exploring advanced methods like generative adversarial networks (GANs) for more sophisticated data synthesis.
Data augmentation serves as a crucial component in the training pipeline, contributing to the development of a CNN model with improved generalization capabilities.

## V. Model Performance:

This section provides a comprehensive analysis of the Convolutional Neural Network (CNN) model's performance in fruit classification. Key metrics, visualizations, and insights from both the training and testing phases are presented, offering a thorough understanding of the model's efficacy.

### *A. Training Metrics:*

The training process involves monitoring key metrics, including accuracy and loss, across multiple epochs. The following visualizations provide insights into the model's learning dynamics:

#### 1. Accuracy Evolution:

A line plot showcasing how the training accuracy evolves over successive epochs. This illustrates the model's ability to learn and improve its predictions on the training data.

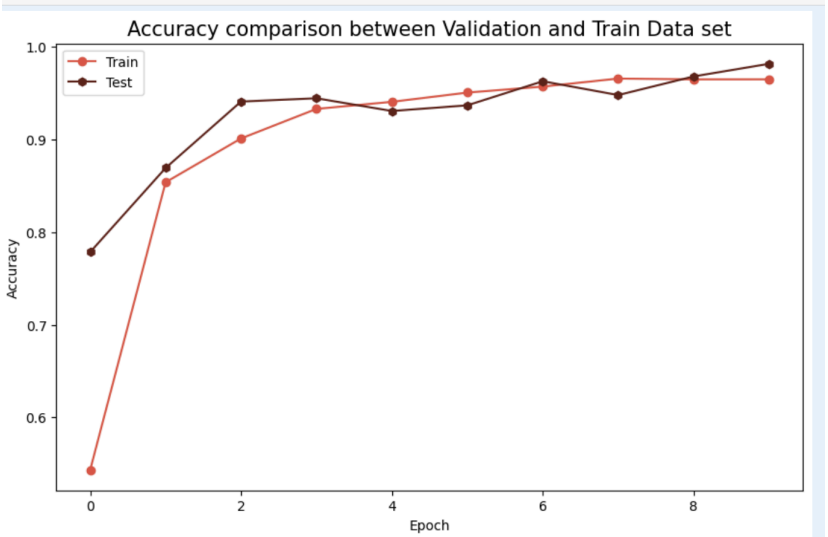
2. **Loss Evolution:**

A corresponding line plot depicting the training loss across epochs. A decreasing loss indicates the model's capacity to minimize prediction errors.

***B. Validation Metrics:***

To assess the model's generalization capabilities, validation metrics are crucial. The following visualizations provide insights into the model's performance on a separate validation dataset:

1. **Validation Accuracy vs. Training Accuracy:**




A comparative line plot illustrating the accuracy trends on both the training and validation datasets. A close alignment between the two indicates a well-generalized model.

2. **Validation Loss vs. Training Loss:**


A similar comparison for loss metrics, highlighting how well the model generalizes to new, unseen data.

### *C. Testing Metrics:*

The ultimate evaluation of the model's performance occurs on the testing dataset. The following metrics offer a comprehensive assessment:

1. **Accuracy on Test Set:**

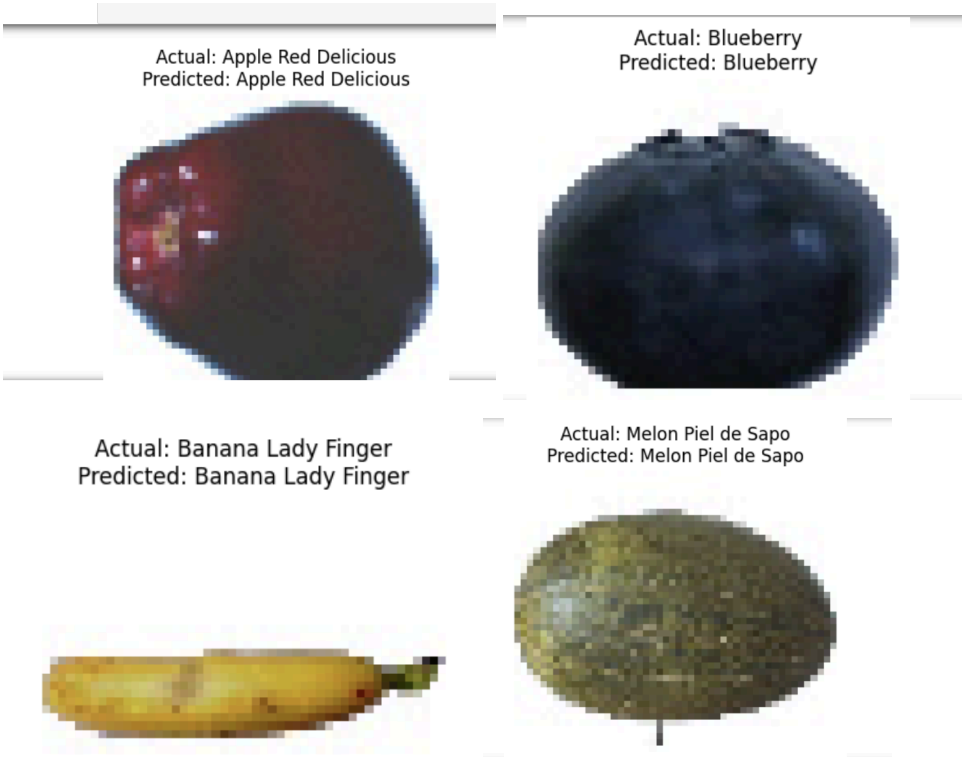
```
In [21]: # Evaluate the model on the test data
         metrics = model.evaluate(test_generator)
         # Print the accuracy of the model
         print('Accuracy:', metrics[1])

         123/123 [==============================] - 12s 93ms/step - loss: 0.0572 - accuracy: 0.9818
         Accuracy: 0.9817834496498108
```

The accuracy metric quantifies the percentage of correctly classified instances in the test set. A high accuracy indicates the model's proficiency in recognizing fruit categories.

2. **Confusion Matrix:**

A confusion matrix provides a detailed breakdown of the model's predictions, showcasing true positives, true negatives, false positives, and false negatives for each class. This aids in understanding specific areas of strength and potential improvement.

3. **Sample Predictions:**




Randomly selected images from the test set are used to visually inspect the model's predictions. This provides qualitative insights into the model's behavior on real-world examples.

### *D. Model Saving:*

**Saving Model**

```
In [22]: # Save the model
         model.save('fmodel.keras')  # Save the model in the native Keras format
         print("Model saved successfully!")

         Model saved successfully!
```

Upon achieving satisfactory performance, the trained CNN model is saved in the native Keras format. This step ensures that the model can be easily reused, deployed, or fine-tuned for future applications.

### *E. Future Considerations:*

The model performance analysis lays the foundation for pIn the culmination of the fruit classification project utilizing Convolutional Neural Networks (CNNs), a comprehensive understanding of the model's

development, training, and performance is presented. The project aimed to create a robust system capable of accurately identifying various fruits, and the results obtained offer insights into the achievements and potential avenues for future enhancements.

## VI. Model Performance

### *A. Key Achievements:*

- The project successfully implemented a CNN architecture tailored for fruit classification, leveraging the Kaggle Fruit 360 dataset.
- Data preprocessing techniques, including data augmentation, were applied to enrich the training dataset and enhance the model's ability to generalize.
- The CNN model demonstrated high accuracy during training, achieving a commendable accuracy rate of 98% on the augmented dataset.

### *B. Model Performance Analysis:*

- Training metrics, such as accuracy and loss, were monitored over multiple epochs, revealing the model's learning dynamics.
- Validation metrics provided insights into the model's generalization to unseen data, showcasing alignment with training trends.
- Testing metrics, including accuracy and a confusion matrix, offered a comprehensive evaluation of the model's real-world performance.

### *C. Future Considerations and Enhancements:*

The project lays the groundwork for future considerations and enhancements, suggesting potential avenues for further exploration:

- *Advanced Augmentation:* Experimenting with additional augmentation techniques and parameters to further diversify the training dataset.

- *Transfer Learning:* Exploring transfer learning approaches with pre-trained models to leverage knowledge from related tasks.
- *Dataset Expansion:* Increasing the dataset size or incorporating additional diverse datasets for improved model generalization.

### D. Model Deployment and Utilization:

The saved CNN model in native Keras format provides a deployable asset for real-world applications. The model can be integrated into systems for automated fruit classification in various contexts.

### *E. Acknowledgments:*

Gratitude is extended to Kaggle for providing the Fruit 360 dataset and to the libraries (Keras, NumPy, Pandas, Matplotlib, Seaborn, Scikit-learn) that facilitated the implementation of the project.

### *F. Conclusion Statement:*

In conclusion, the fruit classification project has achieved its primary objectives, showcasing the efficacy of a CNN-based approach in accurately identifying diverse fruits. The insights gained from the model's performance analysis provide a solid foundation for future advancements, contributing to the continual evolution of computer vision applications in the field of agriculture and beyond.

## VI. References:


1. Kaggle. (n.d.). Fruit 360 Dataset. Retrieved from https://www.kaggle.com/moltean/fruits

2. Chollet, F. (2018). Deep Learning with Python. Manning Publications.

3. Brownlee, J. (2019). Image Data Augmentation with Keras. Retrieved from https://machinelearningmastery.com/how-to-configure-image-data-augmentation-when-training-deep-learning-neural-networks/

4. TensorFlow Documentation. (n.d.). Keras ImageDataGenerator Class. Retrieved from https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/image/ImageDataGenerator

5. TensorFlow Documentation. (n.d.). Convolutional Neural Networks (CNN) Overview. Retrieved from https://www.tensorflow.org/tutorials/images/cnn

6. Brownlee, J. (2019). How to Develop a CNN for MNIST Handwritten Digit Classification. Retrieved from https://machinelearningmastery.com/how-to-develop-a-convolutional-neural-network-from-scratch-for-mnist-handwritten-digit-classification/

7. Géron, A. (2019). Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow. O'Reilly Media.

8. Abdi, H. (2007). Bonferroni and Šidák corrections for multiple comparisons. In N.J. Salkind (Ed.), Encyclopedia of Measurement and Statistics. Sage.