# University Of Palestine

## Software Engineering and Artificial Intelligence

## Software Engineering

**Project Title:**

# System Description and Project Analysis for Fruit Classification System

**Student's Name:**


**Mayar Waleed Nawas**
**120220147**


**Supervisor To:**

**Dr.Mohammed Awad**

**Expert System (LAB)**

**2024 - 2025**

**Contents**

# System Description and Project Analysis for Fruit Classification System

## Introduction

The **Fruit Classification System** is a machine learning-based solution designed to classify images of various fruits. This project leverages artificial intelligence, specifically deep learning techniques, to build an efficient model for identifying fruit images. The system uses a fully connected neural network (FCNN) implemented using **Keras** and trained on the **Fruits 360 Dataset**, which contains labeled images of fruits in multiple categories.

This system is practical for real-world applications such as automated sorting in agriculture, inventory management in supermarkets, and educational tools for recognizing fruits.

## System Description

The Fruit Classification System is an image classification model designed to identify and classify images of various fruits. The system utilizes machine learning techniques, specifically a fully connected neural network built using **Keras**, to process and analyze fruit images. The images are resized to **28x28 pixels** to reduce computational complexity while maintaining classification accuracy.

The system works with the **Fruits 360 Dataset**, which contains a variety of fruit images labeled into different categories. The dataset includes images of the following fruits:

- Apple.
- Banana
- Orange
- Pineapple
- Strawberry
- Watermelon
- Grapes
- Peach
- Lemon
- Pear
- Cherry
- Avocado
- Papaya
- Tomato
- Mang

These fruit categories provide a diverse range of images that the model will learn to classify based on their visual features.
This system is ideal for applications like inventory management in supermarkets, automated sorting in agriculture, or educational tools for identifying fruits. It offers efficient performance and is easy to deploy for small-scale fruit classification tasks.

## Objectives:

### Primary Goals:

1. Develop a machine learning model capable of classifying fruit images based on their visual features.
2. Implement the model using **Keras** and optimize it to work with images resized to **28x28 pixels**.
3. Create a simple friendly interface to demonstrate the classification of fruit images from the Fruits 360 Dataset.

### Secondary Goals:

1. Achieve high accuracy and minimize errors in classification.
2. Ensure scalability for future improvements and larger datasets.

## Functional Requirements:

- **Input:** Images of fruits resized to **28x28 pixels**.
- **Process:**
  - **Preprocessing** the images (resizing and normalization).
  - **Training** the model using a labeled dataset of fruits, including the following categories: Apple, Banana, Orange, Pineapple, Strawberry, Watermelon, Grapes, Peach, Lemon, Pear, Cherry, Avocado, Papaya, Tomato, Mango.
  - **Predicting** the fruit category based on the input image.
- **Output:** The predicted fruit category and the confidence score for the prediction.

## Non-Functional Requirements:

- **Accuracy:** The model should aim for **at least 85% accuracy** in classifying fruit images.
- **Scalability:** The system should be capable of handling larger datasets as it grows.
- **Usability:** The system should display clear and interpretable results for users.

## Dataset Analysis:

- **Dataset**:

  Fruits 360 Dataset containing a variety of fruit images, including **24** categories of fruits such as apple, banana, orange, pineapple, and more.

- **Data Access:**

  To use Kaggle datasets, the user needs to generate a **kaggle.json** file from their Kaggle account and **upload** it to the working environment. Detailed steps are provided in the official Kaggle API documentation: Kaggle API Documentation.

- **Data Preprocessing:**
  - Resize all images to **28x28 pixels** for uniformity.
  - Normalize pixel values to the range **[0, 1].**
  - Split the dataset into **training**, **validation**, and **testing sets** to ensure unbiased evaluation.

## Tools and Technologies Used:

- **Programming Language:** Python
- **Libraries:**
  - **Keras:** To build the neural network.
  - **TensorFlow:** Backend for model training and evaluation.
  - **NumPy:** For data handling.
  - **Matplotlib:** For visualization.
- **Development Environment:** Google Colab for writing and executing the code.
- **Hardware: Google Drive** resources provided by **Google Colab.**

## Environment Setup

- Before training the model, I set up the environment using Google Colab and linked my Google Drive to access the dataset. I uploaded the fruit images to my Google Drive and accessed them via the following code:

join Google Drive at Google Colab

```
from google.colab import drive
drive.mount('/content/new_drive')
```

```
Mounted at /content/new_drive
```

- The images were stored in a specific folder on **Drive**, which was then used to load the dataset for training.
- In addition to the **core libraries and frameworks**, the project employed additional tools like **ipywidgets**, which were used to *create interactive widgets*, and **gdown**, utilized for *downloading files directly from Google Drive*. These tools streamlined the project workflow in the Colab environment

## Development Phases:

### Phase 1: Data Preparation

- **Objective:** Download and preprocess the dataset.
- **Tasks:**
  - Load the Fruits **360 Dataset**.
  - Resize images to **28x28 pixels**.
  - Normalize pixel values to the range **[0, 1]** for efficient processing.
  - Split the data into **training**, **validation**, and **testing sets** to ensure unbiased evaluation.

### Phase 2: Model Building

- **Objective:** Construct a neural network using **Keras**.
- **Tasks:**
  - Define the architecture using **fully connected layers** (**Dense layers**).
  - Train the model on the preprocessed dataset.

### Phase 3: Evaluation and Testing

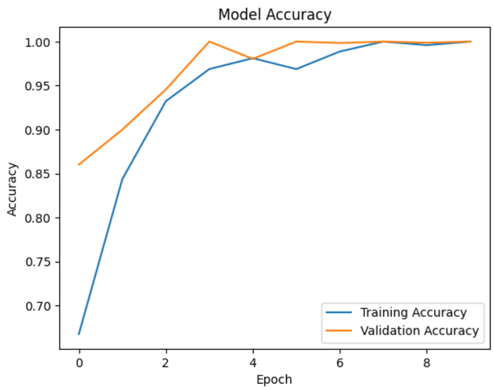
- **Objective:** Evaluate the model performance.
- **Tasks:**
  - Evaluate the trained model on a test set.
  - Fine-tune the model for better performance.

### Phase 4: Deployment

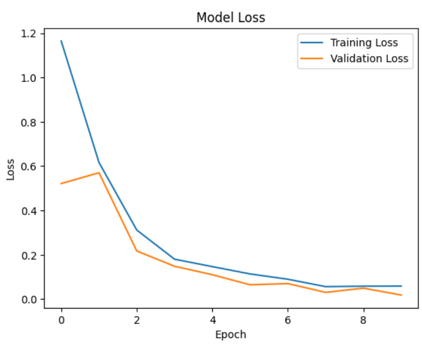
- **Objective:** Create a simple interface for the classification.
- **Tasks:**
  - Create a simple interface to upload and classify fruit images.
  - Display the prediction result and confidence score.

## Performance Metrics :

- **Accuracy**:
  - **Training Accuracy**: The model achieved a training accuracy of **100%**, reflecting its ability to classify fruit images effectively after optimization.
  - **Validation Accuracy**: The validation accuracy reached **100%**, showing excellent generalization during training.
  - **Test Accuracy**: The model achieved a remarkable test accuracy of **99.84%**.




- **Loss**:
  - **Training Loss: 0.0591**, showing minimal error during the training phase.
  - Validation Loss: 0.0189, confirming consistent learning.
  - Test Loss: 0.0460, demonstrating low errors in unseen data.



- **Training Time**:
  - The model was trained for **10 epochs** and completed training in approximately **2 minutes** using **Google Colab's GPU** environment.

❖ Training Accuracy: **100%**

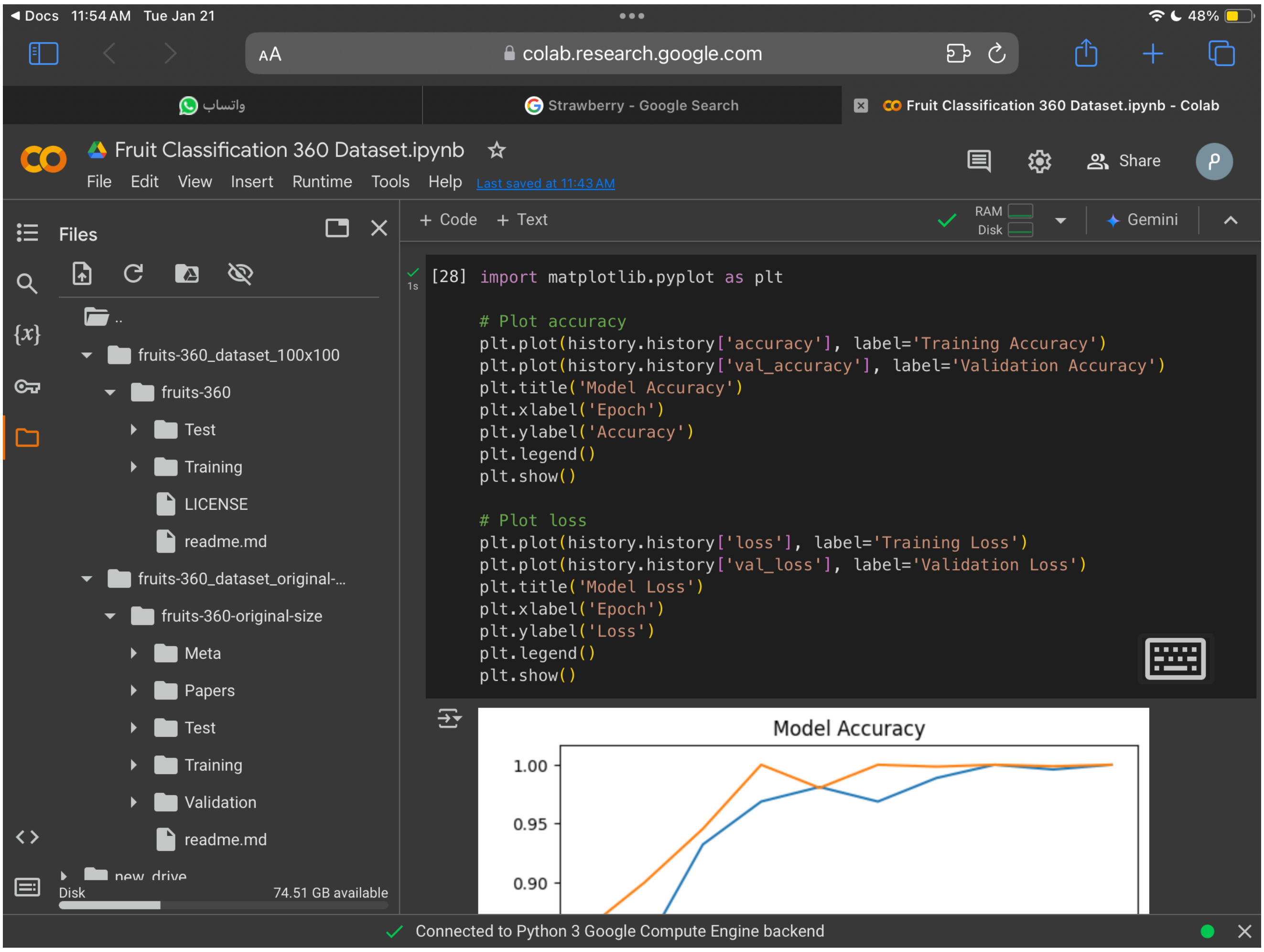
❖ Validation Accuracy: **100%**

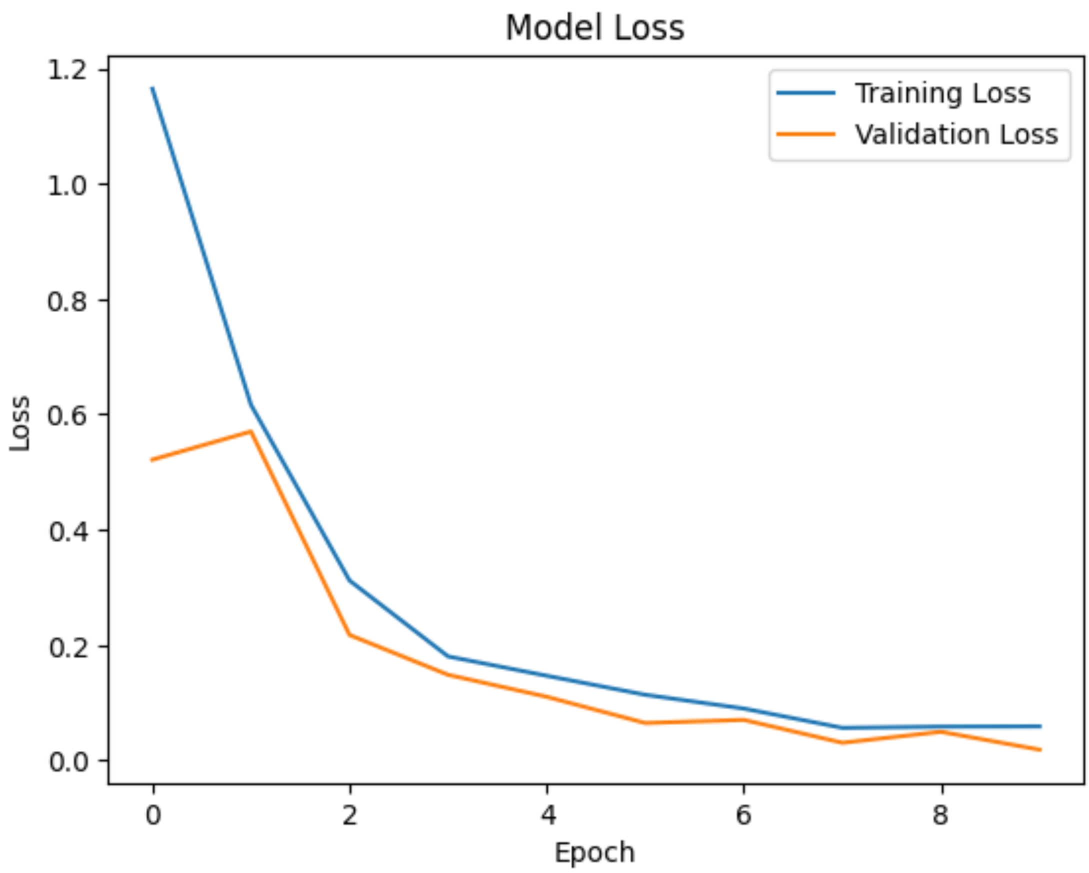
❖ Test Accuracy: **99.84%**

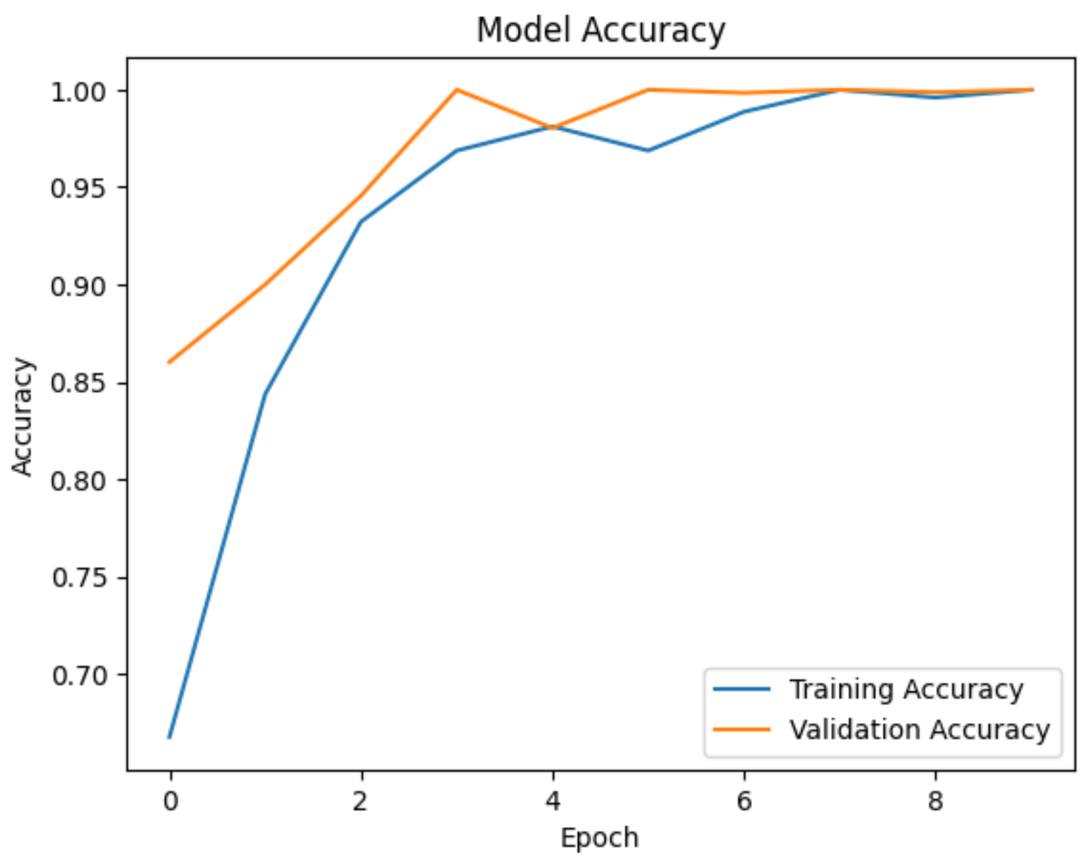
❖ Training Loss: **0.0591**

❖ Validation Loss: **0.0189**

❖ Test Loss: **0.0460**






**Scree**

**nshots and**

# Results :

Copy of Untitled3.ipynb

File Edit View Insert Runtime Tools Help Save failed

Preparing data

```
[14] import os

     # تحديث المسارات
     train_dir = '/content/fruits_dataset/fruits-360_dataset_original-size/fruits-360-original-size/Training'
     validation_dir = '/content/fruits_dataset/fruits-360_dataset_original-size/fruits-360-original-size/Validation'
     test_dir = '/content/fruits_dataset/fruits-360_dataset_original-size/fruits-360-original-size/Test'

     # التأكد من وجود المسارات
     print("Training Directory:", os.listdir(train_dir)[:5])  # عرض أول 5 مجلدات تدريب
     print("Validation Directory:", os.listdir(validation_dir)[:5])  # عرض أول 5 مجلدات تحقق
```

```
Training Directory: ['apple_golden_2', 'pear_1', 'apple_6', 'apple_red_1', 'apple_golden_1']
Validation Directory: ['apple_golden_2', 'pear_1', 'apple_6', 'apple_red_1', 'apple_golden_1']
```

Copy of Untitled3.ipynb

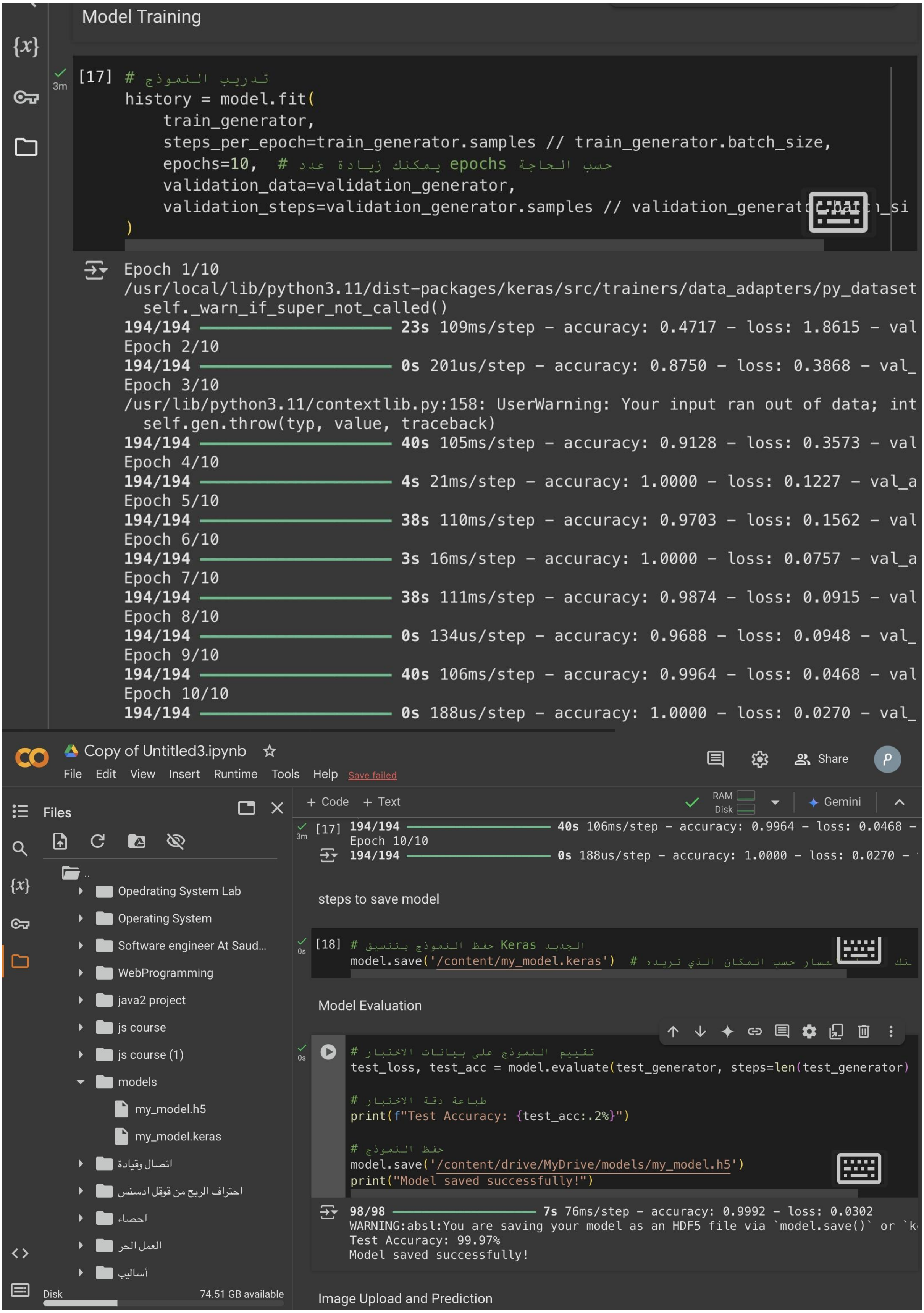
File Edit View Insert Runtime Tools Help Save failed

Training Model with data

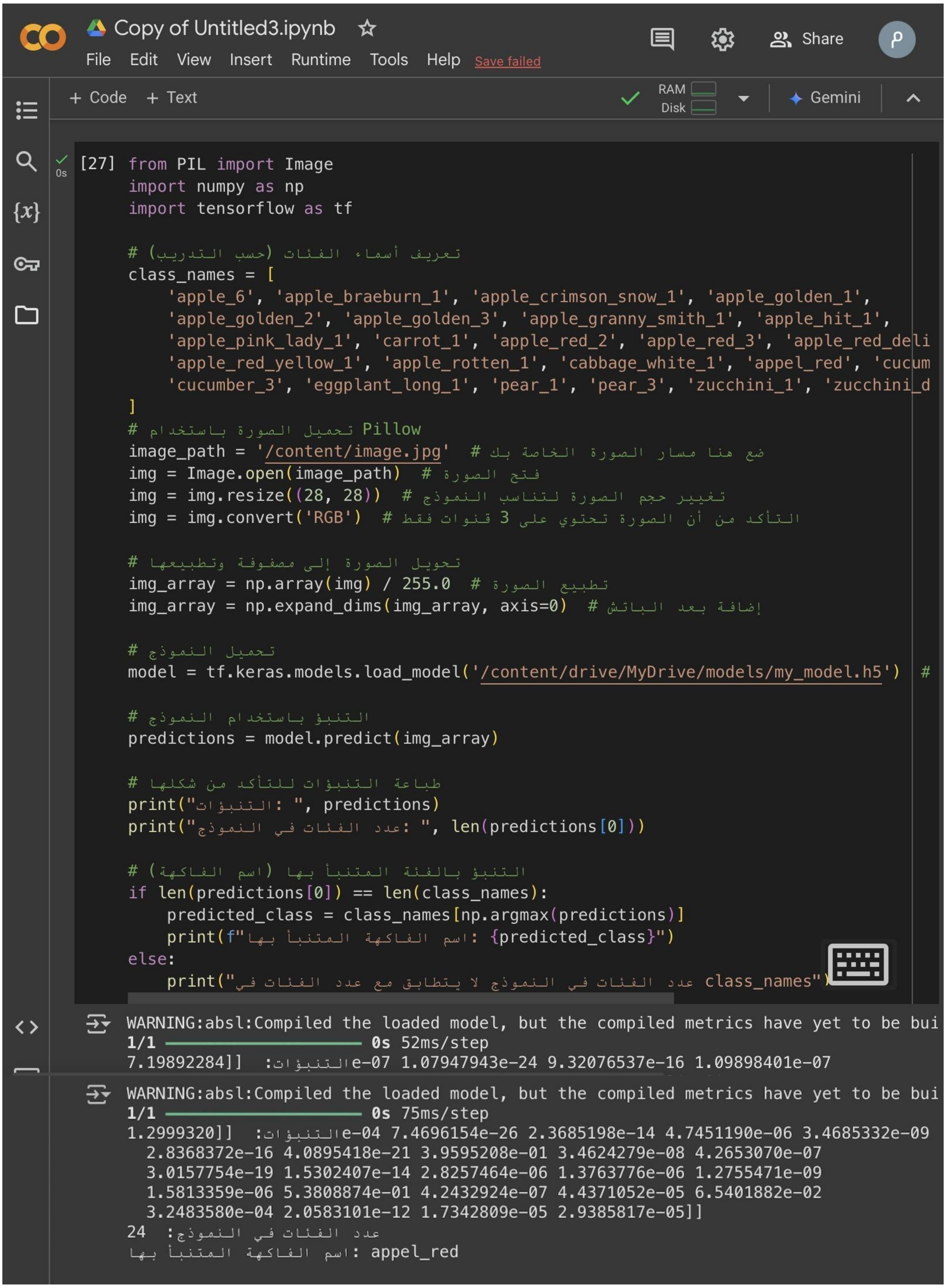
```
[15] from tensorflow.keras.preprocessing.image import ImageDataGenerator

     # إعداد مُولد البيانات
     train_datagen = ImageDataGenerator(rescale=1./255, validation_split=0.2)
     test_datagen = ImageDataGenerator(rescale=1./255)

     # تحميل بيانات التدريب
     train_generator = train_datagen.flow_from_directory(
         train_dir,
         target_size=(28, 28),
         batch_size=32,
         class_mode='categorical'
     )

     # تحميل بيانات التحقق
     validation_generator = train_datagen.flow_from_directory(
         validation_dir,
         target_size=(28, 28),
         batch_size=32,
         class_mode='categorical'
     )

     # تحميل بيانات الاختبار
     test_generator = test_datagen.flow_from_directory(
         test_dir,
         target_size=(28, 28),
         batch_size=32,
         class_mode='categorical'
     )
```

```
Found 6231 images belonging to 24 classes.
Found 3114 images belonging to 24 classes.
Found 3110 images belonging to 24 classes.
```

# Training and Validation Accuracy Graph.

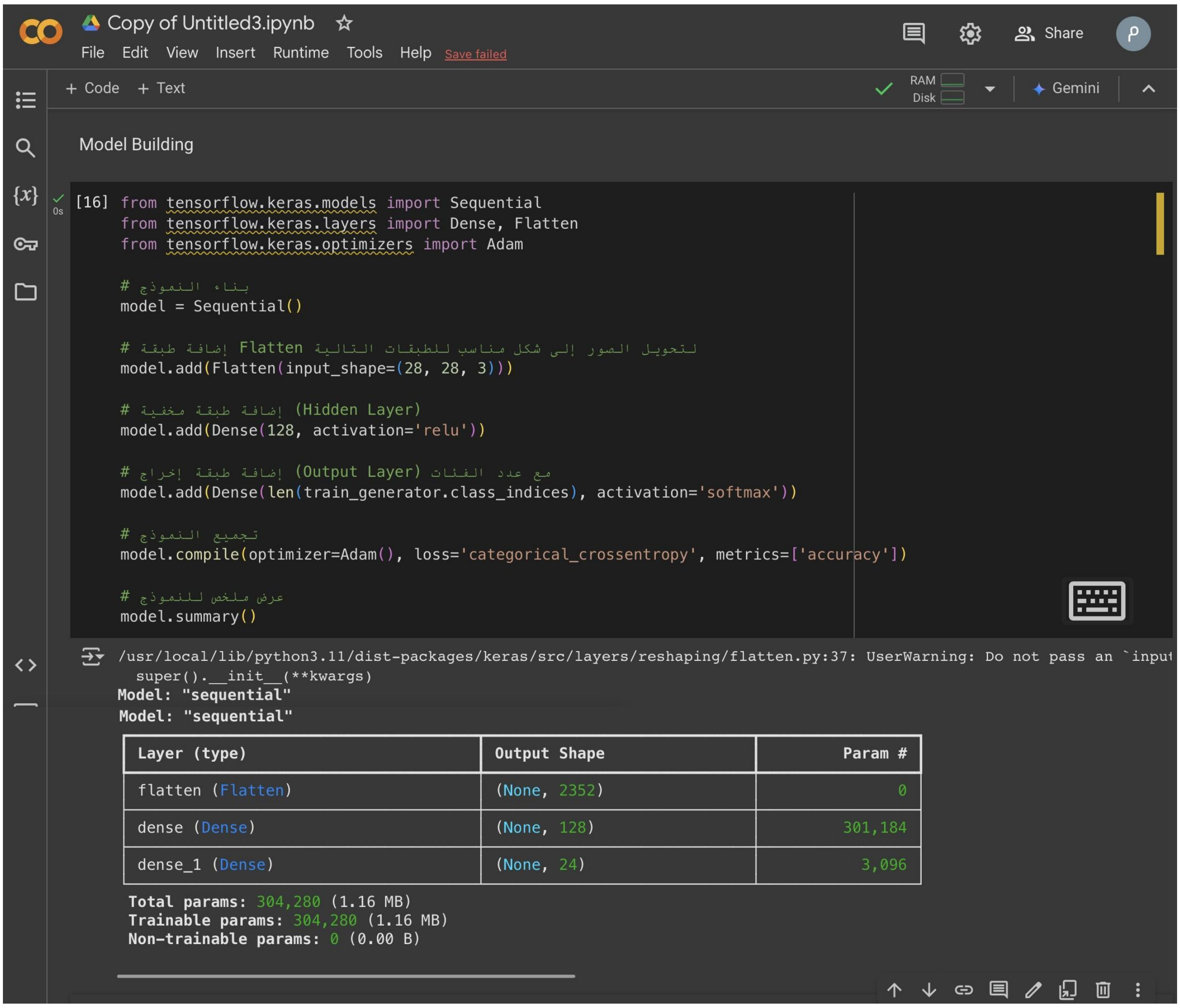


**Sample Input and Output:** Screenshots of the interface with uploaded images and displayed predictions.



# Model Summary:

Output of the model.summary() command, showing the layers and parameters.




```python
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Flatten
from tensorflow.keras.optimizers import Adam

# بناء النموذج
model = Sequential()

# إضافة طبقة Flatten لتحويل الصور إلى شكل مناسب للطبقات التالية
model.add(Flatten(input_shape=(28, 28, 3)))

# إضافة طبقة مخفية (Hidden Layer)
model.add(Dense(128, activation='relu'))

# إضافة طبقة إخراج (Output Layer) مع عدد الفئات
model.add(Dense(len(train_generator.class_indices), activation='softmax'))

# تجميع النموذج
model.compile(optimizer=Adam(), loss='categorical_crossentropy', metrics=['accuracy'])

# عرض ملخص للنموذج
model.summary()
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/reshaping/flatten.py:37: UserWarning: Do not pass an `input
  super().__init__(**kwargs)
Model: "sequential"
Model: "sequential"
```

| Layer (type) | Output Shape | Param # |
|---|---|---|
| flatten (Flatten) | (None, 2352) | 0 |
| dense (Dense) | (None, 128) | 301,184 |
| dense_1 (Dense) | (None, 24) | 3,096 |

**Total params:** 304,280 (1.16 MB)
**Trainable params:** 304,280 (1.16 MB)
**Non-trainable params:** 0 (0.00 B)

## Challenges and Future Suggestions

### Challenges Faced:

- Limited computational resources for training large datasets.
- Optimizing the model to reduce overfitting.

### Future Suggestions:

- Expand the dataset to include more fruit categories and higher-quality images.
- Integrate advanced machine learning techniques like Transfer Learning for better accuracy.
- Deploy the system as a web application for broader accessibility.

## Conclusion:

This modified **Fruit Classification System** leverages the **Fruits 360 Dataset,** which includes a wide variety of fruits for training the machine learning model. The system utilizes **Keras** and a **fully connected neural network** to accurately classify images resized to **28x28 pixels**. This approach ensures efficient performance while maintaining high accuracy in fruit classification tasks.

## References


- Fruits 360 Dataset: Link to Dataset
- Kaggle Platform: Kaggle
- Keras Documentation: Keras
- TensorFlow Documentation: TensorFlow


## Appendices

The **FruitClassificationProjectMayarWaleedNawas120220147** includes the following files**:**

1. **models**
   - **my_model.h5:** The trained model in H5 format.
   - **my_model.keras:** The trained model in Keras format.
2. **docs**
   - **Fruit_Classification_360_Dataset.docx:** The detailed project report.
3. **code**
   - **Fruit_Classification_360_Dataset.ipynb:** Jupyter Notebook containing the code implementation.
   - **fruit_classification_360_dataset.py:** Python script version of the code.
4. **screenshots**
   - Screenshots showing project execution and output results.
5. **data**
   - Sample images used for testing.
6. **PDF**
   - **Fruit_Classification_360_Dataset.pdf**

7. **Video**

   ○ **Part2.docx**

8. **Source Code**
   - The project source code is attached in the appendices for reference.
   - The code can be accessed online through Google Colab at: Colab Link.

9. **Library**

   - **Libraries**
     - ○ **ipywidgets**: Used to create interactive widgets for ***better visualization*** and ***interaction*** during ***training and testing***.
     - ○ **gdown**: Utilized for ***downloading files directly*** from ***Google Drive, streamlining dataset preparation***.
     - ○ The corresponding files for the libraries and their installation guides are located in the "**Library**" folder in the project.

## Additional Notes

"All files and results, including training files and the model, were initially stored on **Google Drive** during the project phases. "

***Thank you for reviewing this project. Your valuable feedback is highly appreciated.***