atomcamp

AgriVision: Automated Multi-Class Fruit Classification and Quality Control System using Deep Transfer Learning

1. Project Overview

In the agricultural industry and food supply chains, automated quality control is essential to reduce waste and ensure safety. Your task is to build a computer vision system capable of classifying fruits.

This project focuses on modernizing agricultural supply chains by developing a robust Computer Vision system capable of identifying distinct fruit varieties. By leveraging **Transfer Learning** (using architectures like MobileNetV2 or ResNet), you will build a scalable, high-accuracy model to automate manual sorting processes, reducing waste and increasing efficiency in food processing lines.

2. Learning Objectives

By completing this project, you will demonstrate the ability to:

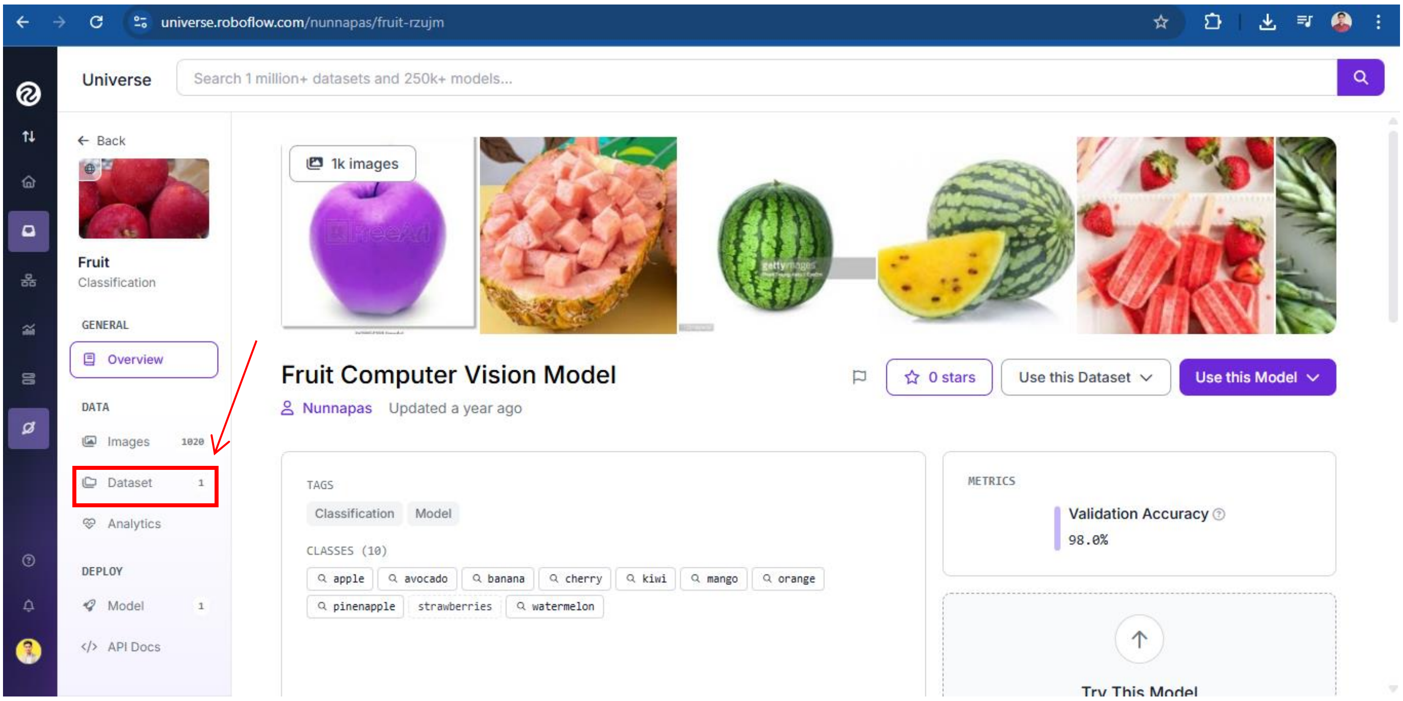
- Load and preprocess image datasets for deep learning.
- Implement data augmentation strategies to improve model generalization.
- Apply Transfer Learning using pre-trained architectures (e.g., VGG16, ResNet, MobileNet).
- Fine-tune models to adapt to specific domain tasks.
- Evaluate model performance using appropriate metrics and visualizations.

3. Dataset Details

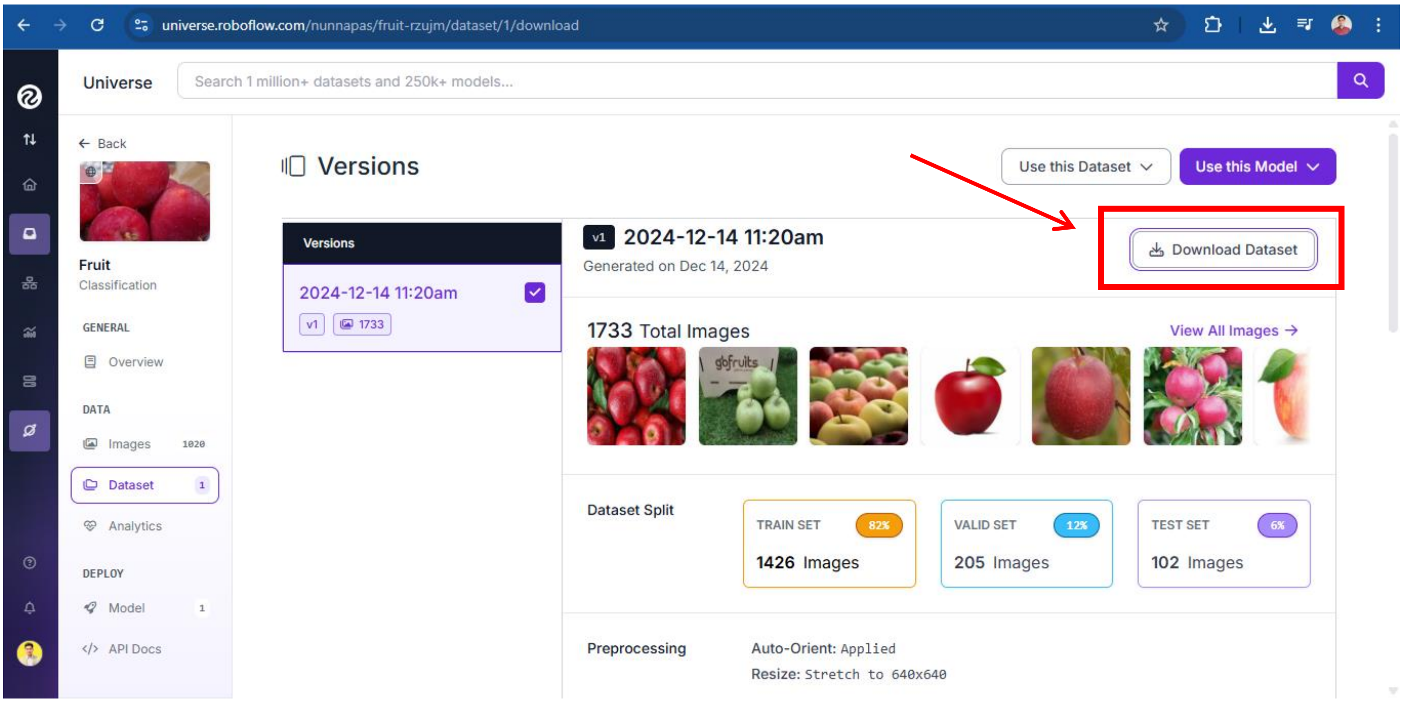
You must use the following dataset for this project. No external datasets are allowed unless explicitly permitted for testing purposes.

- Source: Roboflow - https://universe.roboflow.com/nunnapas/fruit-rzujm
- Classes: The dataset covers 10 types of fruits: Apple, Avocado, Banana, Cherry, Kiwi, Mango, Orange, Pineapple, Strawberries, and Watermelon.
- Structure:
    - test/ (Contains images for evaluating the model)
    - train/ (Contains images for training the model)
    - valid/ (Contains validation data)

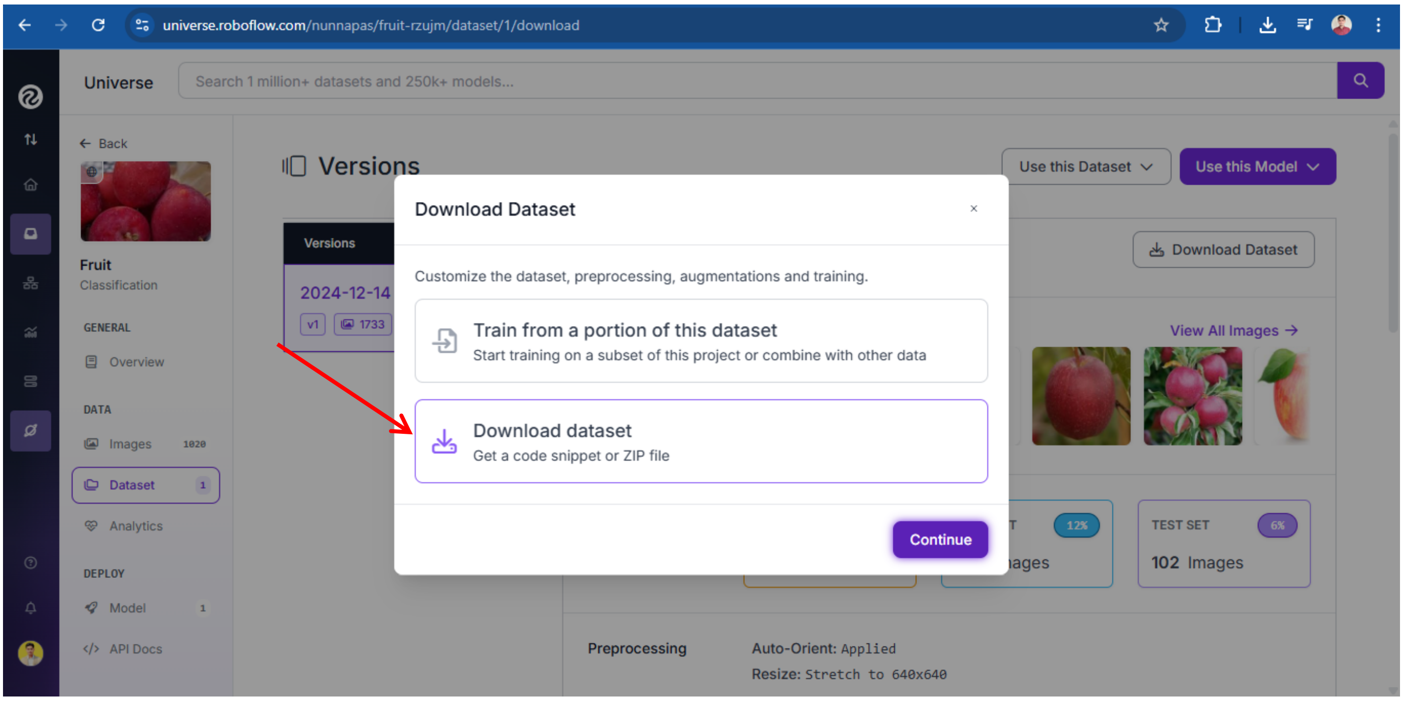
Here is how you can download the data: Click Here > Under Data section select Dataset



Click on Download Dataset Button



Select Download Dataset and Click Continue



Download Dataset

Customize the dataset, preprocessing, augmentations and training.

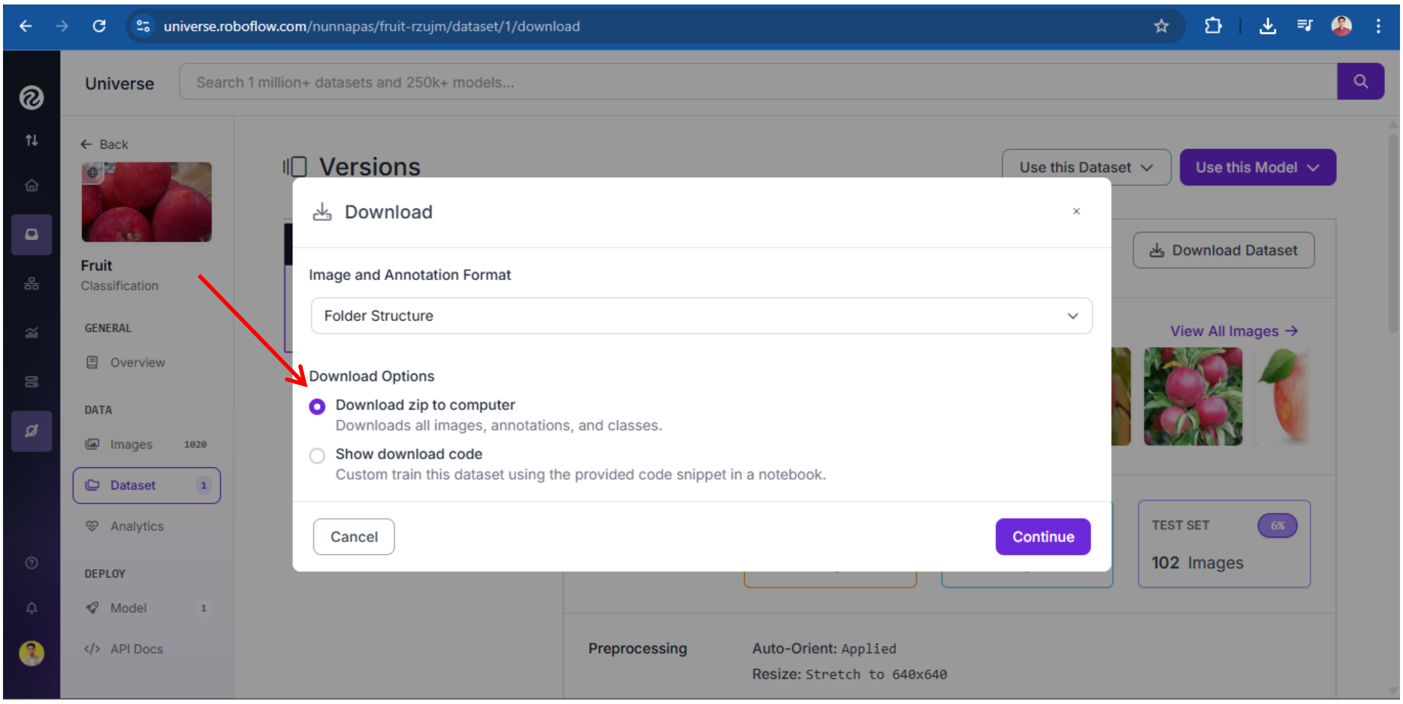
Train from a portion of this dataset
Start training on a subset of this project or combine with other data

Download dataset
Get a code snippet or ZIP file

Continue

Preprocessing: Auto-Orient: Applied
Resize: Stretch to 640x640

Select Download zip to computer and Continue. Hurrah! Check the downloads directory on your machine.



Download

Image and Annotation Format: Folder Structure

Download Options

● Download zip to computer
Downloads all images, annotations, and classes.

○ Show download code
Custom train this dataset using the provided code snippet in a notebook.

Cancel
Continue

Preprocessing: Auto-Orient: Applied
Resize: Stretch to 640x640

- Upload the ZIP file to your Google Drive. Mount your Drive in the notebook (from google.colab import drive) and unzip the dataset programmatically using Python (e.g., !unzip).
- If using Local Jupyter: Extract the ZIP file into your project directory.

4. Project Tasks & Methodology

You are expected to follow the machine learning pipeline described below.

Phase 1: Data Exploration & Preprocessing (15 Points)

1. **Exploratory Data Analysis (EDA):**
   - Identify Classes: Use Python's os library to list the sub-folders in the train directory. Print the list of class names.
   - Visualize Samples: Display one sample image for every class found.
   - Distribution Plot: Plot a bar chart showing the number of images per class in the train set to check for imbalance.
2. **Preprocessing:**
   - Resize images to the input shape required by your chosen model (e.g., 224x224).
   - Normalize pixel values (e.g., rescale to 0-1 range).
3. **Data Augmentation:**
   - Implement data augmentation on the **training set only** to prevent overfitting.
   - *Suggested techniques:* Rotation, Zoom, Horizontal Flip, Width/Height Shift.

Phase 2: Model Development - Transfer Learning (40 Points)

You must implement a robust Transfer Learning solution.

1. **Select a Pre-trained Model:**
   - Choose a standard architecture (e.g., VGG16, ResNet50, InceptionV3, or MobileNetV2) pre-trained on ImageNet.
2. **Feature Extraction:**
   - Load the pre-trained model **without the top** (fully connected) classification layers (often called include_top=false).
   - Freeze the weights of the base layers so they are not updated during the initial training.
3. **Custom Classification Head:**
   - Add your own dense layers with ReLU activation on top of the frozen base.
   - Include Dropout layers to reduce overfitting.
   - **Output Layer:** Must use Softmax activation with *N* units (*N* is number of distinct classes).
4. **Training:**
   - Compile the model using Adam optimizer and CategoricalCrossentropy loss function.
   - Train for enough epochs (minimum 5 or 10).
   - Implement **Early Stopping** to prevent unnecessary computation if the validation loss stops improving.

Phase 3: Fine-Tuning (Optional)

- Once the head is trained, you may unfreeze some of the top layers of the pre-trained base model.
- Re-train the model with a very low learning rate (e.g., 1e-5) to adapt specific features to the fruit dataset.

Phase 4: Evaluation & Analysis (30 Points)

1. **Quantitative Metrics:**
    - Report the final Accuracy on the Test set.
    - Generate a Classification Report (Precision, Recall, F1-Score for each class).
2. **Visualizations:**
    - **Training Curves:** Plot Training vs. Validation Accuracy and Loss over epochs.
    - **Confusion Matrix:** Plot a heatmap of the confusion matrix
3. **Error Analysis:**
    - Display 5 images from the test set where the model made a **wrong prediction**.
    - Provide a brief comment on *why* the failure might have happened (e.g., "The model confused a Red Apple with a Tomato due to similar shape and color").

Phase 5: Deployment / Inference (15 Points)

- Create a simple function predict_fruit(image_path) that takes a local image path as input and prints:
    - The predicted class.
    - Returns the **Predicted Class** and the **Confidence Score** (probability).

5. Deliverables

You must submit the following:

1. **Source Code (.ipynb):** A clean, runnable Google Colab notebook.
    - Code must be well-commented.
    - Output cells (graphs, logs) must be visible (do not clear outputs).
2. **Project Report (PDF):** A 2-3-page summary containing:
    - Introduction & Objective.
    - Methodology (Which model did you choose and why?).
    - Results (Screenshots of graphs, confusion matrix, and metrics).
    - Conclusion (What worked, what didn't?).

End of document