Computational logic for the freshness detection using camera

---

1. Initialization and Setup

Libraries Used

1. **`numpy`**: Used for numerical computations and preparing image data as arrays.
2. **`tensorflow`**: Used to load and run predictions using a pre-trained CNN model.
3. **`matplotlib.pyplot`**: Used for displaying captured images with predictions.
4. **`cv2 (OpenCV)`**: Used for interfacing with the webcam, image capture, and preprocessing.
5. **`os`**: Used for file path operations (not explicitly used in the code snippet but useful for flexible file handling).

---

2. Loading the Model

Code:

```
cnn = tf.keras.models.load_model(model_path)
```

- **Computational Task**:
  - The trained CNN model stored in `.h5` format is loaded into memory.
  - This model contains pre-learned weights and architecture to classify images into predefined categories like freshness and type of fruit/vegetable.
- **Model Details**:
  - Input Shape: `(64, 64, 3)` - Requires images resized to 64x64 pixels with 3 color channels (RGB).
  - Output: A vector of probabilities corresponding to each class in `manual_class_names`.

---

3. Webcam Setup

Code:

```
cap = cv2.VideoCapture(0)
```

- **Computational Task**:
  - Opens the default webcam for real-time image capture.

    - If the webcam fails to initialize, an error message is displayed, and the script exits.
    - We use (0) for default camera and (1),(2) for external camera

User Interaction:

- The user is prompted to press `q` to capture an image for evaluation.

---

4. Image Capture

Code:

```
ret, frame = cap.read()
```

- **Computational Task**:
    - Captures a single frame from the webcam feed.
    - Returns `ret` (Boolean) to confirm successful capture and `frame` (image matrix).
- **Output Data**:
    - `frame`: A matrix of pixel values with shape `(height, width, 3)` in BGR format.

---

5. Image Preprocessing

Code:

```
frame_resized = cv2.resize(frame, (64, 64))
frame_rgb = cv2.cvtColor(frame_resized, cv2.COLOR_BGR2RGB)
input_arr = tf.keras.preprocessing.image.img_to_array(frame_rgb)
input_arr = np.expand_dims(input_arr, axis=0)
```

- **Computational Task**:
    1. **Resizing**:
        - Resize the captured frame to `64x64` pixels, matching the model's expected input shape.
    2. **Color Space Conversion**:
        - Convert the image from BGR (OpenCV default) to RGB.
    3. **Array Conversion**:
        - Convert the RGB image to a NumPy array with pixel values scaled appropriately (e.g., 0-255).
    4. **Batch Preparation**:

- Expand dimensions to simulate a batch of size `1`, as the model expects inputs of shape `(batch_size, height, width, channels)`.

---

6. Prediction

Code:

```
predictions = cnn.predict(input_arr)
result_index = np.argmax(predictions[0])
predicted_category = manual_class_names[result_index]
```

- **Computational Task**:
  1. **Prediction**:
     - The CNN processes the input batch and outputs a probability vector (e.g., `[0.1, 0.3, 0.6]` for 3 classes).
  2. **Class Identification**:
     - Use `np.argmax` to find the index of the maximum probability, which corresponds to the predicted class.
  3. **Category Mapping**:
     - Map the index to a class name using the `manual_class_names` list.

---

7. Metadata Extraction

Code:

```
path_parts = predicted_category.split("_")
freshness_category = path_parts[0] if len(path_parts) > 0 else
"Unknown Freshness"
fruit_info = path_parts[1] if len(path_parts) > 1 else "Unknown
Fruit"
```

- **Computational Task**:
  1. **String Parsing**:
     - The predicted class name (e.g., `fresh_apple`) is split into parts using _ as a delimiter.
  2. **Data Extraction**:
     - The first part (`path_parts[0]`) represents the freshness category (e.g., `fresh`).
     - The second part (`path_parts[1]`) represents the fruit/vegetable type (e.g., `apple`).

8. Visualization

Code:

```
plt.imshow(frame)
plt.title(f"Predicted Freshness: {freshness_category}\nFruit:
{fruit_info}")
plt.axis("off")
plt.show()
```

- **Computational Task**:
    1. **Display the Captured Frame**:
        - The frame is rendered using `matplotlib`.
    2. **Add Prediction Details**:
        - The title includes the freshness category and the fruit type derived from the prediction.

---

9. Releasing Resources

Code:

```
cap.release()
```

- **Computational Task**:
    - Releases the webcam resource, ensuring it can be accessed by other programs.

Improvement Opportunities:

1. **Real-Time Classification**:
    - Instead of capturing a single frame, continuously display predictions in real-time.
2. **Confidence Threshold**:
    - Add a threshold to suppress low-confidence predictions (e.g., `if max(predictions) > 0.8`).
3. **Dynamic Class Names**:
    - Automatically infer class names from the model, reducing the risk of mismatch.

Computation Flow Diagram

1. **Input**: Webcam captures a real-time image.
2. **Preprocessing**:
   - Resize, RGB conversion, and batch preparation.
3. **Model Prediction**:
   - Generate a probability vector and map to a class.
4. **Metadata Extraction**:
   - Derive freshness and fruit/vegetable type from the predicted class name.
5. **Output**:
   - Display the image with the predicted freshness and fruit type.

This logic ensures an efficient pipeline from image acquisition to prediction and visualization. It combines machine learning and image processing techniques to evaluate fruit and vegetable freshness.

Comprehensive computational logic for the freshness detection using image path

---

1. Initialization and Library Imports

Code:

```
import numpy as np
import tensorflow as tf
import matplotlib.pyplot as plt
import cv2
import os
```

Purpose:

1. **`numpy`**: Used for array manipulation and numerical operations.
2. **`tensorflow`**: Loads the trained CNN model and processes input images.
3. **`matplotlib.pyplot`**: Visualizes the test image for inspection and debugging.
4. **`cv2`**: Reads and preprocesses images using OpenCV.
5. **`os`**: Handles file paths dynamically across different operating systems.

---

2. Loading the Pre-Trained CNN Model

Code:

```
cnn =
tf.keras.models.load_model(r'C:\Users\lenovo\Downloads\trained_m
odel5.h5')
```

Purpose:

- Loads the `.h5` file containing the pre-trained CNN model. This model was previously trained to classify images into categories like freshness levels and fruit types.

Assumptions:

- The model was trained on images resized to `(64, 64)` pixels and normalized as expected by TensorFlow.

---

3. Preparing the Test Dataset

Code:

```
test_set = tf.keras.utils.image_dataset_from_directory(
    test_dir,
    labels="inferred",
    label_mode="categorical",
    image_size=(64, 64),
    batch_size=32
)
```

Purpose:

- Loads test images from the specified directory into a TensorFlow dataset.
- **Key Parameters**:
  - `labels="inferred"`: Automatically infers labels from folder names.
  - `label_mode="categorical"`: Labels are one-hot encoded
  - (e.g., `[0, 1, 0, ...)` for multi-class classification).
  - `image_size=(64, 64)`: Resizes all images to match the input size required by the model.
  - `batch_size=32`: Processes images in batches for efficiency.

Output:

- `test_set`: A batched dataset of test images and corresponding labels.

---

4. Test Image Visualization

Code:

```
img = cv2.imread(image_path)
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
plt.imshow(img)
plt.title('Test Image')
plt.xticks([]), plt.yticks([])
plt.show()
```

Purpose:

- Reads a specific test image from the filesystem and displays it using `matplotlib`.

- **Steps**:
  1. Reads the image using OpenCV (`cv2.imread`).
  2. Converts the image from OpenCV's default BGR format to RGB for correct color representation in `matplotlib`.
  3. Displays the image with a title for visual inspection.

---

5. Extracting Metadata from the File Path

Code:

```
path_parts = image_path.split(os.sep)
freshness_category = path_parts[-3]
fruit_info = path_parts[-2].split("(")[0].strip()
shelf_life = path_parts[-2].split("(")[1].split(")")[0].strip()
if "(" in path_parts[-2] else "Unknown"
```

Purpose:

- Extracts metadata (e.g., freshness category, fruit type, and shelf life) from the test image's directory structure and filename.

Steps:

1. **Split Path**:
   - Uses `os.sep` to split the file path into components dynamically (system-agnostic).
   - For example: `C:\Users\lenovo\Desktop\ezyzip\test\FRESH (75-100% FI)\apple(7-14 DAYS)\image.jpg`.
2. **Extract Freshness Category**:
   - Assumes the folder name two levels above the image contains freshness info (e.g., `FRESH (75-100% FI)`).
3. **Extract Fruit Type**:
   - Uses the folder directly above the image (e.g., `apple(7-14 DAYS)`) and extracts the fruit name (e.g., `apple`).
4. **Extract Shelf Life**:
   - Parses text within parentheses in the folder name for shelf life details (e.g., `7-14 DAYS`).

---

6. Preprocessing the Image for Model Prediction

Code:

```
image = tf.keras.preprocessing.image.load_img(image_path,
target_size=(64, 64))
input_arr = tf.keras.preprocessing.image.img_to_array(image)
input_arr = np.array([input_arr])
```

Purpose:

Prepares the image for model inference by following the preprocessing pipeline used during training.

Steps:

1. **Load Image**:
   - Uses TensorFlow's `load_img` to load the image from disk and resize it to `(64, 64)` pixels.
2. **Convert to Array**:
   - Converts the image to a NumPy array with pixel values scaled between `0` and `255`.
3. **Batch Preparation**:
   - Adds an additional dimension to simulate a batch (shape: `(1, 64, 64, 3)`).

---

7. Model Prediction

Code:

```
predictions = cnn.predict(input_arr)
result_index = np.argmax(predictions[0])
predicted_category = test_set.class_names[result_index]
```

Purpose:

- Performs prediction on the preprocessed image and retrieves the predicted class.

Steps:

1. **Prediction**:
   - The CNN outputs a probability vector (e.g., `[0.1, 0.3, 0.6]`).
2. **Class Identification**:
   - Uses `np.argmax` to find the index of the highest probability, which corresponds to the predicted class.
3. **Class Mapping**:
   - Maps the predicted index to the corresponding class name using `test_set.class_names`.

Output:

- `predicted_category`: The predicted freshness category (e.g., `FRESH (75-100% FI)`).

---

8. Displaying Results

Code:

```
print(f"Predicted Freshness Category: {predicted_category}")
print(f"Fruit Type: {fruit_info}")
print(f"Remaining Shelf Life: {shelf_life}")
```

Purpose:

- Outputs the results to the console for user review:
  1. **Predicted Freshness Category**: Model's prediction.
  2. **Fruit Type**: Extracted from the file path.
  3. **Remaining Shelf Life**: Extracted from the file path.

---

9. Summary of Workflow

1. **Model Loading**:
   - The pre-trained CNN is loaded for inference.
2. **Test Data Preparation**:
   - A test dataset is prepared, and a single test image is visualized for verification.
3. **Metadata Extraction**:
   - Freshness category, fruit type, and shelf life are parsed from the file path.
4. **Image Preprocessing**:
   - The test image is resized and formatted to match the CNN's input requirements.
5. **Prediction**:
   - The CNN predicts the freshness category.

6. **Results Display**:
    - Outputs the predicted category alongside metadata.

This logic ensures consistency, accuracy, and efficiency in evaluating the freshness of fruits/vegetables using a trained CNN model.