SmartHome Gesture Control Application Project Part 2 Report

1. Introduction

The goal of this project is to build a gesture recognition application for SmartHome devices. The application processes gesture videos by extracting representative frames, obtaining feature vectors from a pre-trained Convolutional Neural Network (CNN), and then classifying the gesture using cosine similarity.

2. Approach

- **Extracting the Middle Frame:**
  The extract_middle_frame function captures the middle frame of each video file, as this frame is most likely to represent the gesture adequately.
- **Unified Feature Extraction:**
  The new function get_penultimate_features(folder) handles both training and test data. It:
    - Iterates over video files in a given folder.
    - Extracts and converts the middle frame to grayscale.
    - Uses the HandShapeFeatureExtractor to extract the penultimate layer features.
    - For training videos, it parses the filename to extract a gesture name using the delimiter _PRACTICE_ and then maps this to a numerical label via training_gesture_mapping.
    - For test videos, it extracts the gesture name using a hyphen (-) delimiter and appends the name as the label for CSV reporting.
- **Classification via Cosine Similarity:**
  The recognize_gestures function computes the cosine similarity between each test feature vector and all training feature vectors. The training sample with the highest similarity determines the predicted gesture.
- **Output Generation:**
  The predicted labels, along with the corresponding gesture names (extracted from the test filenames), are written to a CSV file (Results.csv) that includes two columns: "Gesture Name" and "Output Label."

3. Implementation and Solution

The solution is implemented in a modular Python script (main.py) with the following key functions:

- **extract_middle_frame(video_path)**
  Captures the middle frame of a video, which is used as the representative frame for feature extraction.
- **get_penultimate_features(folder)**
  A unified function that processes a folder of videos. For each video:
    - It extracts the middle frame.

    - Converts the frame to grayscale.
    - Uses the CNN model (via HandShapeFeatureExtractor) to obtain the feature vector.
    - Depending on whether the folder is "traindata" or "test", it extracts the gesture name from the filename using the appropriate delimiter and maps it (for training data) or directly stores the name (for test data).
- **compute_cosine_similarity(vec1, vec2)**
  Calculates the cosine similarity between two feature vectors.
- **recognize_gestures(test_features, train_features, train_labels)**
  Compares each test feature vector to the training feature vectors using cosine similarity, then assigns the gesture label of the training sample with the highest similarity.
- **CSV Output:**
  The main function writes the results to a CSV file named Results.csv with two columns:
    - **Gesture Name:** Extracted from the test video filename.
    - **Output Label:** The predicted gesture label based on cosine similarity.

4. Evaluation of the Application

This implementation successfully integrates multiple components:

- **Modular and Unified Code:**
  By using a single extraction function for both training and test videos, the code is simpler and easier to maintain.
- **Effective Feature Representation:**
  Leveraging the penultimate layer of a pre-trained CNN ensures that the feature vectors are rich in information for distinguishing between different gestures.
- **Robust File Parsing:**
  Custom parsing logic for training and test filenames allows the system to correctly map gesture names to numerical labels despite differences in naming conventions.
- **Accurate Classification:**
  The use of cosine similarity for comparing feature vectors has proven effective for recognizing gestures.
- **Clear Output Format:**
  The CSV output with two columns provides an organized and easily interpretable result that matches project specifications.

Potential Improvements:

- Further pre-processing (e.g., image normalization or enhancement) could improve the feature extraction process.
    - Details in gesture may vary due to different arm position, etc.
    - Extracting at the middle frame of the video might not be the most accurate.
- Additional experiments with different similarity metrics or classification strategies might further enhance accuracy.

5. Conclusion

The project demonstrates a successful approach to gesture recognition for SmartHome control. By systematically extracting features from gesture videos and leveraging cosine similarity for classification, the application meets the project requirements. The modular design, along with unified feature extraction and robust file parsing, lays a strong foundation for future enhancements. This solution effectively bridges deep learning-based feature extraction and classical similarity-based classification, offering a scalable approach for gesture-based SmartHome applications.