SmartHome Gesture Recognition Application

Part 2 – Gesture Recognition Using Deep Feature Similarity

Name: Beneil Sourisho Chamaky
Course: CSE 535 – Multimedia and Web Databases
Project: SmartHome Gesture Control Application – Part 2

---

1. Introduction

The objective of this project is to recognize hand gestures from video input for a SmartHome control system. The task is to use a pre-trained convolutional neural network (CNN) to extract discriminative handshape features from gesture videos and identify unknown gestures using cosine similarity.

Unlike traditional classification approaches, this project emphasizes feature-based similarity matching using the penultimate layer of a CNN model. This allows gesture recognition without retraining the network, relying instead on similarity comparisons between known training gestures and unseen test gestures.

---

2. Problem Understanding

The problem consists of two datasets:

- **Training Dataset:**
  A set of 51 gesture videos recorded by the user in Part 1, covering:
  - Numeric gestures (0–9)
  - SmartHome commands (FanOn, FanOff, FanUp, FanDown, LightOn, LightOff, SetThermo)
- **Test Dataset:**
  A provided dataset of 51 unseen gesture videos.

The goal is to:

1. Extract a representative frame from each video.
2. Pass the extracted frame through the penultimate layer of a pre-trained CNN.
3. Compare test feature vectors against training feature vectors using cosine similarity.

4. Output the predicted gesture number for each test video.

---

3. Approach

The overall workflow closely follows the provided project flowchart and is summarized below.

3.1 Frame Extraction

For both training and test videos, a middle frame is extracted from each video. This frame is assumed to best represent the gesture while avoiding transitional frames at the beginning or end of the video.

Steps:

- Load the video using OpenCV
- Compute the total number of frames
- Select the middle frame
- Convert the frame to grayscale

This ensures consistent input format for the CNN.

3.2 Feature Extraction Using CNN

A pre-trained CNN model (cnn_model.h5) is provided. Instead of using the final classification layer, the penultimate layer output is used as a feature vector representing the handshape.

Key points:

- Images are resized to 200×200 pixels
- Pixel values are normalized
- The CNN outputs a high-dimensional feature vector
- No retraining or modification of the model is performed

A singleton class (HandShapeFeatureExtractor) is used to ensure the model is loaded only once into memory.

3.3 Training Feature Vector Construction

For each training video:

1. Extract the middle frame
2. Generate a feature vector using the CNN
3. Store the feature vector along with its gesture label (derived from filename)

This produces a feature database representing known gestures.

3.4 Test Feature Vector Construction

The same process is applied to each test video:

1. Extract middle frame
2. Generate feature vector using the same CNN
3. Store the feature vector for similarity comparison

This ensures that training and test data are processed identically.

3.5 Gesture Recognition Using Cosine Similarity

To identify the gesture in each test video, **cosine similarity** is computed between the test feature vector and all training feature vectors.

Cosine similarity is defined as:

$$\text{cosine similarity} = \frac{A \cdot B}{\| A \| \| B \|}$$

The training gesture with the **highest similarity score** (or lowest cosine distance) is selected as the predicted gesture.

3.6 Output Generation

Each predicted gesture is mapped to its corresponding numeric ID based on the project specification:

Gesture	ID
Num0–Num9	0–9
FanDown	10
FanOff	11
FanOn	12
FanUp	13
LightOff	14
LightOn	15
SetThermo	16

The final predictions are written to a file named Results.csv, containing exactly 51 lines, one integer per test video.

---

4. Solution Summary

The implemented solution successfully follows the required workflow:

- Used provided CNN model without modification
- Extracted penultimate layer features

  Applied cosine similarity for gesture recognition
- Processed both training and test datasets uniformly
- Generated numeric gesture outputs in required format

This approach avoids retraining and demonstrates how deep feature embeddings can be effectively reused for similarity-based recognition tasks.

---

5. Conclusion

This project demonstrates a practical application of deep learning feature extraction combined with similarity-based classification. By leveraging a pre-trained CNN and cosine similarity, gesture recognition can be performed efficiently even with limited training data.

The implemented system adheres strictly to the project guidelines and successfully produces valid gesture predictions for all test videos. This approach highlights the effectiveness of deep feature representations for multimedia recognition tasks and provides a scalable solution for SmartHome gesture control systems.