Ira A. Fulton Schools of Engineering
Arizona State University

CSE 535: Mobile Computing

SmartHome Gesture Control Application: Part 2

Purpose

Gesture-based SmartHome devices have the ability to increase convenience, but also create more accessible SmartHome devices for the elderly and those with disabilities. In this project, you will develop an application service that will control SmartHome devices with gestures, and then develop a RESTful application service for classifying the SmartHome gestures. This project provides hands-on experience developing a mobile application and provides an excellent opportunity to gain further exposure to topics and applications in the areas of mobile computing and machine learning.

Objectives

At the completion of this individual project, learners should be able to:

- Develop a python application that classifies specific gestures
- Apply pre-trained machine learning model usage for classification
- Train and test a CNN model

Description

To complete Part 2 of the project, you will develop a Python application for classifying SmartHome gestures. You will gain hands-on experience in training and testing a CNN model for hand gestures, and applying pre-trained machine learning model usage for classification.

Use the practice gesture videos generated in project Part 1 and test gesture videos provided in the *test.zip* file and source code provided. The zip file is available in your Coursera course in the *Course Project Overview* page in the *Welcome and Start Here* section.

SmartHome Gesture Control Application: Part 2 is auto-graded. The overall project accounts for 40% of your grade.

Technology Requirements

- TensorFlow==2.12.0

- Python 3.10
- OpenCV for Python(4.8.0.74)
- Keras==2.12.0
- Pandas==1.5.3
- Numpy==1.24.3

Directions

Functionality of the Application

Task 1: Generate the penultimate layer for the training videos.

Steps to generate the penultimate layer for the training set:

1. Extract the middle frames of all the training gesture videos.
2. For each gesture video, you will have one frame extract the hand shape feature by calling the get_Intsance() method of the HandShapeFeatureExtractor class. (HandShapeFeatureExtractor class uses CNN model that is trained for alphabet gestures)
3. For each gesture, extract the feature vector.
4. Feature vectors of all the gestures is the penultimate layer of the training set.

Task 2: Generate the penultimate layer for the test videos

Follow the steps for **Task 1** to get the penultimate layer of the test dataset.

Task 3: Gesture recognition of the test dataset.

Steps for gesture recognition of the test dataset:

1. Apply cosine similarity between the vector of the gesture video and the penultimate layer of the training set. Corresponding gesture of the training set vector with minimum cosine difference is the recognition of the gesture.
2. Save the gesture number to the "Results.csv"
3. Recognize the gestures for all the test dataset videos and save the results to the "Results.csv" file.

Gesture Name	Output Label
0	0
1	1
2	2
3	3
4	4

5	5
6	6
7	7
8	8
9	9
Decrease Fan Speed	10
FanOff	11
FanOn	12
Increase Fan Speed	13
LightOff	14
LightOn	15
SetThermo	16

Submission Directions

Deliverables

For Part 2 of the project, you must submit **five (5)** deliverables in a **single zip file**.

1. Save the python application as "main.py"
2. Save the training gesture videos under the "traindata" folder.
3. README file (optional)
4. Save the results (recognized gestures) to the "Results.csv" file.
5. A report evaluating your application that discusses the following points:
    a. An explanation of how you approached the given problem
    b. Solution for the problem

Important notes about your deliverables:

- You can have as many auxiliary python files as you want, but the autograder will **only** run the "main.py", and it should generate the "Results.csv". The generated "Results.csv" file should **not** include any headers and should only contain integers in 51 x 1 matrix where each row represents your predicted label.
- While generating the "Results.csv" file in "main.py", assume the file should be generated in the root directory to be picked up by the autograder. Do **not** add an absolute path inside "main.py" while submitting your solution file.

- The test data in the autograder will be placed in the "test" folder inside the working directory as provided in the source code.

Zip File

Your submission must be a **single zip file** that contains your five (5) project deliverables. The zip file should **only** contain these items, and it must follow this naming convention:

- *CSE535Project2_(full name).zip*

Do not place your deliverables in a folder and then zip that folder. Please follow this folder hierarchy list for your submission:

Submission Project Folder

|---- traindata

|--- main.py

|-- cnn.h5

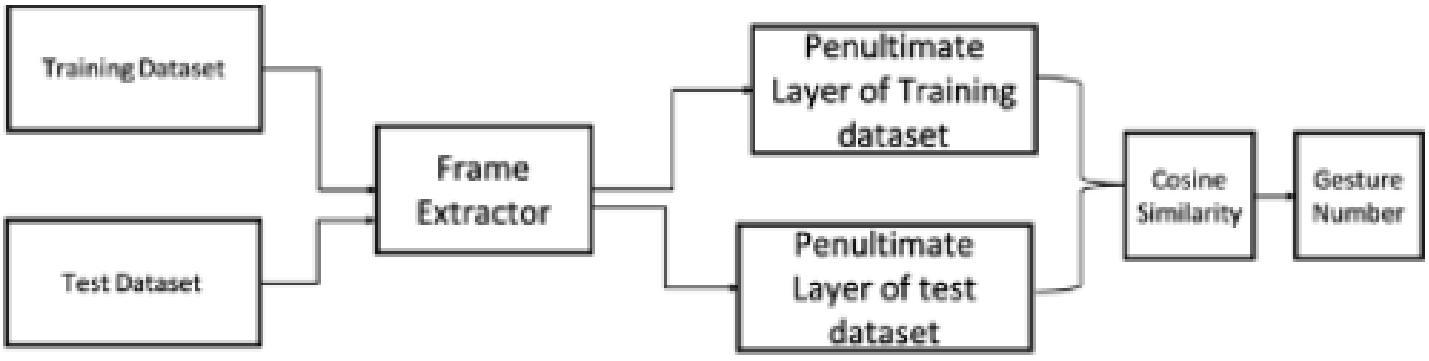
|-- handshape_feature_extractor.py

|-- any additional files you may create

|---- README.md (optional)

|---- Report.pdf

Workflow for the project:



Evaluation

Your submission will be auto-graded. Please review the Evaluation Criteria for how your source code will be assessed. Submissions will be evaluated based on the criteria and will receive a total score.

Submissions missing any part of the project will be graded based on what was submitted against the evaluation criteria. Missing parts submitted after the deadline will **not** be graded.

Review the course syllabus for details regarding late penalties.

Evaluation Criteria

Your submission will be evaluated and scored based on these criteria:

- Compilation errors: 0
- Application Compiles: 85
- Total score: 85 + Accuracy Score
    - "Accuracy Score" is computed by comparing Gesture true labels and user output labels.