Ira A. Fulton Schools of Engineering
Arizona State University

CSE 535: Mobile Computing

SmartHome Gesture Control Application
Project 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

Technology Requirements

- Python 3.10
- Tensorflow 2.15.0
- Keras 2.15.0
- Pandas 2.2.3
- Scikit-learn 1.5.2
- Numpy 1.24.3
- OpencV for Python 4.10.0.84

Project 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 the "Project Overviews and Resources" page in the *Welcome and Start Here* section of your course.

SmartHome Gesture Control Application Project Part 2 is auto-graded.

Directions

Before Starting

It is recommended to create a **free GitHub account**.

Creating a GitHub Account

A GitHub account is needed to submit your deliverables for the course. Please create a free account by following these directions:

1. Navigate to https://github.com/ and click "Sign Up".
2. Use your **asurite@asu.edu** email address to create your account.
3. Create a password for your account.
4. Under "Enter a username", use your asurite username (e.g., sparky123).
5. Follow the verification process when prompted.
6. Select to create a "Free" account when prompted.
7. Create a new repository in your account (refer to Steps to create a new GitHub repository).
    - **Required**: Select "**private**" visibility for your repository.
8. Commit your code to a branch in the repo.

For more details on GitHub refer to https://docs.github.com/en

Clone your repository locally to begin working on the project:

1. On your GitHub, select the repository you created.
2. Select the “Code” button. There are multiple ways to clone the repository:
    a. Cloning with HTTPs URLs
    b. Cloning with SSH URLs
    c. Cloning with GitHubDesktop

3. After cloning to your IDE, you can work on the project and push your code to the remote GitHub repository (refer to Push your code to GitHub).

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.

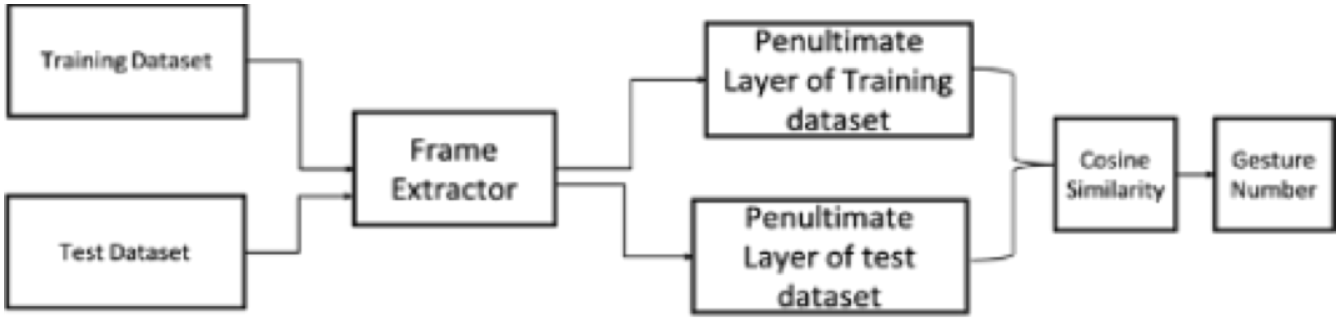
Steps for gesture recognition of the test dataset:

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

Workflow for the Project



The image is a **flowchart diagram** showing the process of gesture recognition using training and test datasets. Here's a step-by-step breakdown:

1. **Inputs:** The dataset consists of video files representing various gestures, organized into two distinct sets:
    - Training Dataset
    - Test Dataset
2. **Frame Extractor:**
    - Both datasets are passed into a **Frame Extractor**, which presumably extracts key video frames or features from the datasets.

3. **Feature Extraction:**
    - The output of the frame extractor is processed through the **Penultimate Layer of Training Dataset** and **Penultimate Layer of Test Dataset**.

4. **Similarity Comparison:**
    - The outputs from both penultimate layers are compared using **Cosine Similarity**, which measures the similarity between two vectors.

5. **Output:**
    - The gesture from the training set that has the highest similarity score is identified as the result.

Submission Directions for Project Deliverables

You must submit your SmartHome Gesture Control Application Project Part 2 deliverables through Gradescope. Carefully review submission directions outlined in this overview document in order to correctly earn credit for your work. Learners may not email or use other means to submit any assignment or project for review, including feedback, and grading.

The SmartHome Gesture Control Application Project Part 2 includes five (5) deliverables in total, one (1) of which is optional and four (4) which must be loaded into your GitHub repository:

1. **README file (optional)**: You may include this file but it is not required.
2. **Training Gesture Videos:** Save the training gesture videos under the "traindata" folder.
3. **Python Application:** Save the Python application as "main.py".
4. **Results CSV File**: Save the results (recognized gestures) to the "Results.csv" file.
5. **Report PDF:** Your report must evaluate your application and should be saved as a single PDF.
    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.

Please follow this folder hierarchy list for your submission in GitHub:

Submission Project Folder
| -- traindata
| -- main.py
|-- cnn.h5
|-- handshape_feature_extractor.py
|-- any additional files you may create
| -- README.md (optional)
| -- Report.pdf

Submitting to Gradescope

After completing work in GitHub, you must submit your work into Gradescope to receive credit for the course:

1. Go to the Canvas Assignment, "**Submission: SmartHome Gesture Application Control Project Part 2**".
2. Click the "**Load Submission…in new window**" button.
3. Once in Gradescope, select the assignment titled "**SmartHome Gesture Application Control Project Part 2**" and a pop-up window will appear.
4. In the pop-up:
    a. You will see the submission method as "GitHub".
    b. If prompted: Authorize your GitHub account with Gradescope (one time only) by clicking "Connect to GitHub".
    c. In your repository section, select the repository which has your submission files.
    d. In the branch section, select the correct branch where your project files are present.
    e. Click "Upload" to submit your work for grading.

5. If needed: to resubmit the assignment in Gradescope:
    a. Return to the Canvas submission and open Gradescope.
    b. You will be navigated to the "Autograder Results" page (if it is not your first submission).
    c. Click the "**Resubmit**" button on the bottom right corner of the page and repeat the process from Step 3.

Your submission will be reviewed by the course team and then, after the due date has passed, your score will be populated from Gradescope into your Canvas grade.

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: 119
- Total score: 119 + Accuracy Score
    - "Accuracy Score" is computed by comparing Gesture true labels and user output labels.