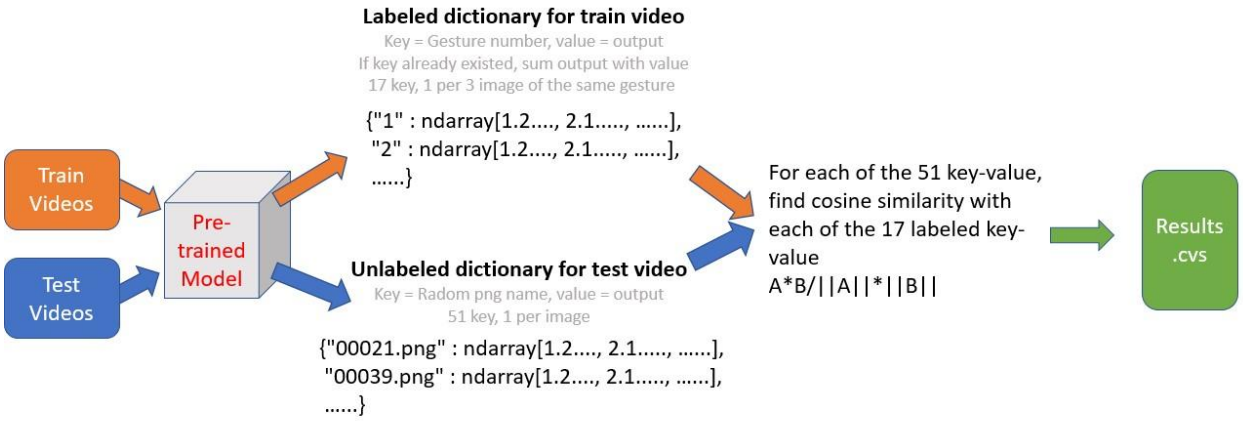
Jiahui Li
SmartHome Gesture Application Control - Part 2 Report



An explanation of how you approached the given problem:

I will discuss my approach to the given program in the order of three parts where I divided the project.

My first task focused on extracting the video. With the given frameextractor class, this part was relatively easy. The videos are passed into the frameExtractor method iterating through the video directory, each of the images is saved into the extracted_training_image folder. I modified the frameextractor class to return the image name after extraction in order to keep track of the trained video label. The image name and video name is put into a key-value dictionary. Example:

The second task is to extract the gesture feature from the image. The first problem occurred is that the test video frame extraction image is flipped upside down. The image is rotated 180 degrees before passing into the HandShapeFeatureExtractor. The training video output from HandShapeFeatureExtractor is stored in a key-value dictionary as value, and later output will be added to the existing value, 17 in total. Example: And the testing video output from HandShapeFeatureExtractor is stored individually with just the random generated .png file name, 51 in total. The third task is to compare each of the 51 stored testing video output to all 17 labeled training video output by using the cosine similarity function. A key-value dictionary was used to keep track of the result. The testing video .png file name was stored as key, and the

value will be updated with the closest label and the cosine similarity. Example: {'00046.png': ['16', 0.9612638]}

Solution for the problem:

The goal is to achieve a high accuracy score computed by comparing Gesture true labels and user output labels. I first compared each of the 51 training videos and the 51 testing videos and tried to find the closest cosine similarity with both the student train cnn model and the included cnn model in extracting the gesture feature. And the included cnn model seems to do a better performance in this case with an accuracy of 5 out of 51, by 5 times compared to the student train cnn model which has an accuracy of 1/51. But the result was still low, then I tried to take the average for the each of the 17 gesture into one output and compare testing video to the 17 averaged output, and this give me a better result with the included cnn mode accuracy increased to 11/51 and the student trained model accuracy increased to 3/51. To check each of the developed algorithms, I ran the same videos into the code twice as both training and testing videos. And the accuracy of the averaged output algorithms seems to perform the best with 81% of accuracy.