Smart Home Project Part II

1st Mateo Ortega
Arizona State University
Computer Science and Engineering Program
Tucson, Arizona
morteg51@asu.edu

***Abstract*—In this project, my primary objective was to train, tests, and validate a Machine Learning model. The focus was on utilizing gestures that could be used in a smart home settings such as controlling devices like lamps. The methodology involved employing Python as the programming language and a series of libraries such as the TensorFlow framework, numpy, and pandas. These tools where used to collect, categorize,create predictions, and export the data into a CSV format. Our findings revealed an 40 percent accuracy rate, which is notably high given the small dataset size of 51 frames since only the middle frames where recorded. This achievement holds significant implications, particularly in creating a better smart home experience for the elderly given their limited accessibility, allowing them to control various appliances by just using gestures.**

I. Introduction

This project is a Machine Learning implementation. This implementation, uses videos recorded by users in the app developed in Part I, which I will go over in another section, and the videos include a specific amount of gestures that users should record. The videos purpose is to create an seamless experience for smart home users, and while anyone can use these gestures it's target audience is mostly the elderly.

II. Problem

The problem includes creating a portable programming environment, gathering all of the data, getting predictions and recording all of the data. A major complication is extracting data, this is because videos have multiple frames in which they contain a variance of data within a video, this means that you have to be consistent with gathering these frames. Another problem is getting predictions can be not accurate given the algorithm used to get those predictions, in the next sections it will be discussed how these problems were resolved.

III. Procedures

A. Setting up the project

First I had to create a Python3.10 environment since the python version I have in my workstation is 3.12 and that is not compatible with Tensorflow. After creating the python virtual environment I installed all dependencies such as Tensorflow, keras, pandas, scikit-learn, numpy, and open-cv. Tensorflow is to create the machine learning model and to be able to do complex math on Tensors. Keras and sciKit-learn is used to make predictions in our machine learning algorithms. Numpy is used to create N by N size arrays in C, this allows for our Python algorithm to be performing. Pandas has many uses, mostly dataframes but in our case was used to write the data gathered in a CSV file. Lastly Open-CV was used to hold the video in memory and gather frames from the videos recorded.

Identify applicable funding agency here. If none, delete this.

B. Storing Data

The first thing I did when the directory is have a place to store all of the videos for testing and training the machine learning algorithms. First I created a train data folder in which it had all 51 training data videos recorded by me using my application and my phone connected to the application. These videos contained all of the gestures with 3 samples per video. After placing all of the training videos, I went ahead and created the Test folder for storing all of the testing videos, this was the same exact way just that they were all videos recorded by me. I also loaded a pretrained machine learning model provided by the professor called gestures trained cnn model.

C. Gathering the labels

My first step on programming was first using the OS library included in the Standard library set from Python, and gathering all of the path file names from our train data and test data folders, this is because of frame extractor function used in the next procedure requires all of the paths to the videos. While gathering all of the training data paths I also extracted the labels for each video and stored them in a Hash-map along with the trial number. This was useful for recognizing which data type each one is. This was all done by taking advantage of the concise formatting and using string manipulation to extract the data. For the test data all that was gathered was the path since no labels are required unless you are trying to measure accuracy.

D. Extracting Frames

First I imported the frame Extractor function from a function created by the professor. This function includes getting the video from the video path and putting in memory for video manipulations. Then the program gets the amount of frames in the video and gets the middle frame in the video based on the length of the video. After getting the frame the programs writes the frame to a folder created at the beginning of the function and writes it to a specific label that is passed on. At the end of the function it returns frame. I created a wrapper function to it which enables storing all of the frames along with the labels in a hash-map for easy of use.

E. Gathering predictions

I imported a functions provided by the professor called hand shape feature extractor which contains a singleton class along with a function that loads the trained model gestures trained CNN model. Now with this function I had to modify the function to remove the color of the image so that the model could take read the images and make proper predictions. The program first iterates through the hash-map that contains all of the frames from the training data, and starts to make predictions, and stores them in another hash-map with the same labels as the key items. Next the program start to create predictions on the array that contains all of the test data and stores it in an array.

F. Classifying data using Cosine Similarity

I created a class that borrows the training prediction's library and stores the training map as an attribute. After that there is two private methods called get y which get's two numpy arrays from each prediction and converts them into a true and a prediction tensor by using the tensorflow convert to tensor function. After collecting both variables the program normalizes them by using the tensorflow nn l2 normalize, after normalizing those tensors they get stored in a member function called y true normalize and y pred normalize. Now the next function is called cosine similarity, which uses the tensorflow reduce sum function along with the tensorflow multiply function with y true normalize and y pred normalize as parameters in the function and this returns a numpy array with a prediction value inside of it. Now the last function in the program is the match closest result which takes in the prediction from the test data, the iterates through all of the predictions of the training data along with the label and compares them all with the test data, and returns the closest cosine similarity value along with the label.

1) Dificulties with Cosine Similarity: The problem with the algorithm is that you have to iterate through each value for each label meaning that if the data was to grow the classification would be very time consuming while improving the accuracy it would also be very slow. Another problem was chosing the approach to do the cosine similiraty since there are many resouces that should different way to do the cosine similarity. Since tensorflow is already a machine learning framework it made the most sense to use it to calculate the cosine similarity.

G. Writing to CSV

Now for the last part I created a hash-map that aligns the label along with a value from 0-17 and then writes it down to a results array. This array will contain all of the closest predictions made from the classification. After putting all of the data in the array the array get's stored in a pandas dataframe, and then gets written down to a results.csv file returning a 51x1 matrix.

IV. Analysis

The project's purpose was made to be as accurate as possible only utilizing a noble amount of data, so it comes with it's limitations. As seen the results from the testing accumulates to an overall 40 percent accuracy, this may be because of the small sample size, the variance in the data, the lighting in the videos, the video quality in between videos. Many cameras pick up color differently, even though I removed the color it still impacts the image quality.

V. Limitations

The problem of creating a machine learning applications is the data. The data collected from the testing can vary. This is generally a good thing but since the training data is a small set, it can be difficult to use it against data that can vary. Things that varied from the testing data where the camera quality, the frame gathered from the video is usually not the ideal one, meaning that you would either have to collect multiple frames from the training data, or get more videos in the training data. The math provided from websites required to use very expensive compute for the cosine similarity.

VI. Future Improvements

This project could improve by getting more data, the more data that can be feed to a model the better. Adding deep learning techniques to the algorithm can also improve the accuracy of the model, this advance artificial intelligence concept has hit strides and can improve the efficiency along with the accuracy of a model. Another improvement that can be done is using a NVIDIA gpu to accelerate the speed of training of the machine learning model, since more data would mean more time. Another improvement that could benefit this project would be to test it in the mobile phone, by giving the user the ability to tell the phone which videos are accurate and which are not, this technique is also called reinforced learning.

VII. Conclusion

This project was about creating a machine learning model to identify different gestures from our videos that were recorded on part I. From the procedures it can be seen that the project was successful as a proof of concept because even with a small sample of videos the model can accurately predict the difference between hand gestures, which is difficult because there are different hand sizes, different hand shapes, different movements in hands between people. The portability of the project can also be applauded since the model can be used in mobile devices because of it's small data size and can be used to even train further the algorithm.