Name: Jason Zhou
Mentor: Dr. Dongjin Song
Status Report #: 13
Time Spent on Research This Week: 2.75
Cumulative Time Spent on Research: 85
Miles Traveled to/from Mentor This Week: 0
Cumulative Miles Traveled to/from Mentor: 0
==========

Monday, December 13th, 2021: (0.25 Hrs)

To begin the week, I had my weekly Zoom meeting with my mentor. Besides discussing some topics like reading files for code, we talked about scheduling. Specifically, he told me that now that the exam periods are mostly over for him, he would probably have more time to respond to emails and meet to discuss my ARM project. Additionally, we also decided to move our weekly meeting for next week to Tuesday instead of Monday (just for that one week).

Saturday, December 18th, 2021: (1 Hr)

On this day, I focused on trying to start the first part of my project, which was to actually get the audio files (the data) into my coding app so that I could reformat them into a usable form. Now, just for context, with just the developmental and additional developmental data, there are over 40,000 files. As such, one could say that I was motivated to find a way to automate the process of inputting the files into my coding format. Upon researching methods for doing this, I discovered that some people use the “glob” function to achieve this, which simply finds files that exhibit a specified pattern (in this case my files all have the same file type) and groups them into an array. Although this is useful, it does not work very well when all of the files are split up into different files. Thus, I will have to find a workaround.

Sunday, December 19th, 2021: (1.5 Hrs)

As previously mentioned, I needed to find a workaround to handle cases when the files are split up within multiple files. However, after scavenging the internet for a solution, it appeared that no one had an appropriate way to handle the problem I was having. As a result, I developed (or tried to develop) my own code to do this.

```
def recurseFiles(path, array):
    if len(glob.glob(path+'\\*.wav')) != 0: #base condition to break out of recursion
        return glob.glob(path+'\\*.wav') #if there is a wav file present, then reteurn/break

    folders = glob.glob(path+'\\*') #if there are no files, then there must be only folders inside the pathway

    for folder in folders: #iterate through all of the folders
        array = np.append(array, recurseFiles(folder, array)) #recurse: rinse and repeat

    return array #returns a numpy array that contains all file addresses for all wav files in dataset
```

(This is a picture of the function I created to take in multiple files across multiple folders.)

The function takes in a file pathway that holds all of the files plus an array that will store all of the files (which is later returned/outputted). This function mainly works by using recursion. In other words, the function calls itself and infinitely runs until a certain condition is met.

The function seems to at least achieve the simple task of collecting a massive number of files. There is just one issue: while testing, I discovered that it seems to store copies of already collected files, which means that the duplicate files end up in the returned array, leading to a higher number of files/repeated data. This is most likely from iterating through a folder too many times, which means that the for-loop would be the culprit.

Looking into the problem, I came across an article about the glob function that (miraculously) described looking through files recursively, which was exactly what I was trying to do. Upon reading further, I realized that the glob function actually had a built-in recursion feature, which happened to solve my exact problem (and made the function I had created completely useless).

```
path2 = 'C:\\Users\\jason\\OneDrive\\Desktop\\DCASE Data\\**\\*.wav'
data = glob.glob(path2, recursive = True)
```

(A picture of the code that solved all my problems and made my previous function useless. This is much easier to implement, as it essentially only uses one line if you exclude the file pathway address.)

In hindsight, I should have spent more time looking for a solution to my problem. Although it is unfortunate that my function is useless, now I have a way to efficiently obtain all the addresses into my code so that I may load them into my coding environment. I can now reformat them into a usable form, which is what I will most likely be working on next week.

References

Sam, S. (2018, October 3). *Glob() function in PHP*. Tutorialspoint. Retrieved December 20, 2021, from https://www.tutorialspoint.com/glob-function-in-php#:~:text=The%20glob()%20function%20returns,directories%20matching%20a%20specified%20pattern.&text=An%20array%20containing%20the%20matched,FALSE%20on%20error.

sid779. (2020, April 25). *How to use glob() function to find files recursively in Python?* GeeksforGeeks. Retrieved December 20, 2021, from https://www.geeksforgeeks.org/how-to-use-glob-function-to-find-files-recursively-in-python/