Focus Test Implementation

# **Introduction**

One of the main motives of our project is to identify a biomarker for focus and feed it to the corresponding ML model. To achieve this, it becomes necessary to design a test that induces the test subject to enter a mode of focus. We have designed and implemented a test that attains this aforementioned goal.

The test includes a very simple exercise. The subject is made to select his/her favorite animal at the beginning of the test. After that, the subject encounters a GUI or graphical user interface where the image of an animal is shown in the background. The image keeps on toggling after very short intervals of time. There is a button at the bottom of the GUI. The person is supposed to click the button when his favorite animal appears on the screen. The process forces our subject to be focused at he/she cannot afford to miss the click when the desired animal appears on the screen. The order in which the images appear is random, thus making it unpredictable for the subject.

Since biomarker identification is our primary motive of the experiment, it is important to record physiological signals while the experiment is being carried out. We make sure that the EEG and PPG data of the subject is logged while the subject is taking the test. This has to be done in a parallel manner as the data logging and the toggling of images have to be performed simultaneously.

# **Tools Used**

* **Tkinter:** For the implementation of the GUI or the Graphical User Interface
* **Random:** To jumble up the ordering of animal pictures from the target folder
* **Serial:** To regulate the serial port that allows us to log the corresponding EEG and PPG data.
* **Threading:** To carry out simultaneous operations that include image toggling and logging of physiological data.
* **PIL:** To read image data from the target folder

# **Variable Description**

**count:** To store and control the index of each animal image as it appears on the screen

**our\_images:** Stores the images in the folder as an array of images

**correct:** To keep count of the correct responses by the subject

**wrong:** To keep count of the wrong responses by the subjects

# **Function Definition**

**log\_data:** The function which reads the data from the serial port line-by-line. After obtaining the data, it decodes it and removes the unnecessary characters in the line. Then the data is logged in an array for later analysis.

**nexta:** This function is responsible for toggling the images in the GUI after fixed intervals of time. The **count** variable is updated in this function. It assumes values randomly within a fixed range according to the size of the **our\_images** array. This function is evoked repeatedly after short intervals of time that allows the background image of the canvas to vary.

**answer:** This function is primarily responsible for validating the response provided by the subject. The **correct** and **wrong** variables are both updated in the process. For each correct response, the **correct** variable is incremented by one and for every incorrect response the **wrong** variable is incremented by one. The **count** also varies as the image is changed the moment our subject chooses to click the button.

# **Program Overview**

The program primarily deploys an end-to-end Focus Test for the subjects. The program involves uses an external folder for image storage to fetch the questions to the GUI. The responses of the subjects are validated and stored. The end time for the test is not extensively defined and the duration of alteration of images is set at 1 sec initially. The GUI displays a single button to accept the response of the subject. Since the program requires the continuous data logging and the toggling of images to be done simultaneously, the concept of threading has been used. Two threads are defined, one for each of the tasks.