THE OHIO STATE UNIVERSITY

CREATE an E-GAMING ENVIRONMENT to INCREASE ATTENTION SPAN of PEOPLE HAVING ADHD (ATTENTION DEFICIT HYPERACTIVITY DISORDER)

M.S. FINAL PROJECT REPORT

Submitted by

Agnibh Dey
Master of Science in Electrical and Computer Engineering
The Ohio State University, Columbus

Under the Guidance of

Dr. Kevin Passino
Professor, Electrical & Computer Engr.
Department of Electrical and Computer Engineering
The Ohio State University, Columbus

INDEX TO CHAPTERS

ABSTRACT

This project is a multidisciplinary initiative with an objective to develop an e-gaming environment to increase the attention span of people with ADHD (Attention Deficit Hyper-Activity Disorder). Conventionally, medications have proved to be very effective to treat ADHD over the short term, but researches have failed to show long-term benefits. In addition, there are significant side effects of medication on children such as hallucinations, tics, weight loss, depression, etc [9].

This project aims to create a machine learning model that can give an indication of the attentiveness of a person and use the predicted parameter to create a training environment that changes according to the person's attention. For this, an attention aware system was designed based on EEG signals to identify continuous attention levels of a person. Then a machine learning model was created based on gaussian regression processes to predict real-time attention. The final step was to simulate a game whose parameters change according to attention levels of the player; lower attention level corresponds to more assistance and fewer challenges compared to a period of high attention

The entire project was divided into four steps:

(i). Developing a task that gives an indication of the attention of a person through EEG (Electroencephalogram) signals,

(ii). Processing the EEG signals to correlate with the attention of the person,

(iii). Creating a model with the data collected,

(iv). Training the person with ADHD on in e-gaming environment to improve his attention span.

SECTION 1: DEVELOPING A TASK TO MEASURE ATTENTION

The first task was to give a number to indicate the attention of the person so that a machine learning model can be created to map the EEG signals to the attention score. For this, a task was developed modelled on TOVA (Test of Variables of Attention) [1] in MATLAB.

The TOVA stimuli consist of two graphic images that are randomly displayed on screen for 100 ms. The images are classified in two sets: Target and Non-target. Target stimuli have the rectangle on the top while the non-target stimuli have the rectangle at the bottom.

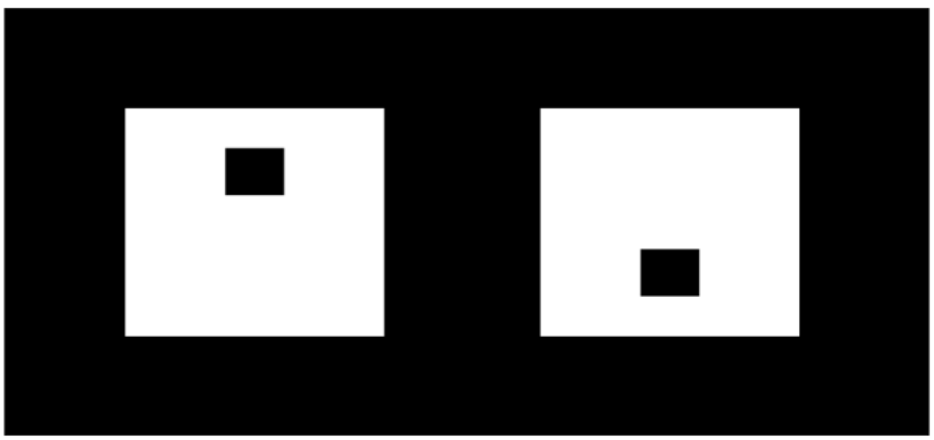

Figure 1: TOVA stimuli- Target and Non-Target

When the target is displayed on screen, then the person performing the task should press the SPACE key and when non-target appears on the screen then the person should press the ENTER key. The reaction time and the key pressed by the person are obtained using the 'getkey' command in MATLAB [2].

An attention score is given to the person based on his performance on the task in the range of 0-5, 0 being a person is least attentive and 5 being a person is most focused on the task. The formula used for calculating the attention score is given as –

$$Scr = \frac{cTrgCrc * cTrgCrcTm + cNTrgCrc * cNTrgCrcTm}{cTrgCrc + cNTrgCrc + 1}$$

$$-2 * \frac{cTrgNcrc * cTrgNcrcTm + cNTrgNcrc * cNTrgNcrcTm}{cTrgNcrc + cNTrgNcrc + 1}$$

where,

cTrgCrc – Count for number of times the correct key is pressed when target appears.

cTrgCrcTm – Sum of the reaction times when correct key is pressed when target appears.

cNTrgCrc – Count for number of times the correct key is pressed when non-target appears.

cNTrgCrcTm – Sum of the reaction times when correct key is pressed when non-target appears.

cTrgNCrc – Count for number of times the wrong key is pressed when target appears.

cTrgNCrcTm – Sum of the reaction times when wrong key is pressed when target appears.

cNTrgNCrc – Count for number of times the wrong key is pressed when non-target appears.

cNTrgNCrcTm – Sum of the reaction times when wrong key is pressed when non-target appears.

Based on the interval in which the variable *'Scr'* lies, a discrete attention score is given to the person-

Scr	**Attention Score**
<1	0
>=1 && <1.6	1
>=1.6 && <2.2	2
>=2.2 && <2.8	3

>=2.8 && <3.4	4
>=3.4	5

The scores are generated at an interval of 10 seconds while the person is performing the task.

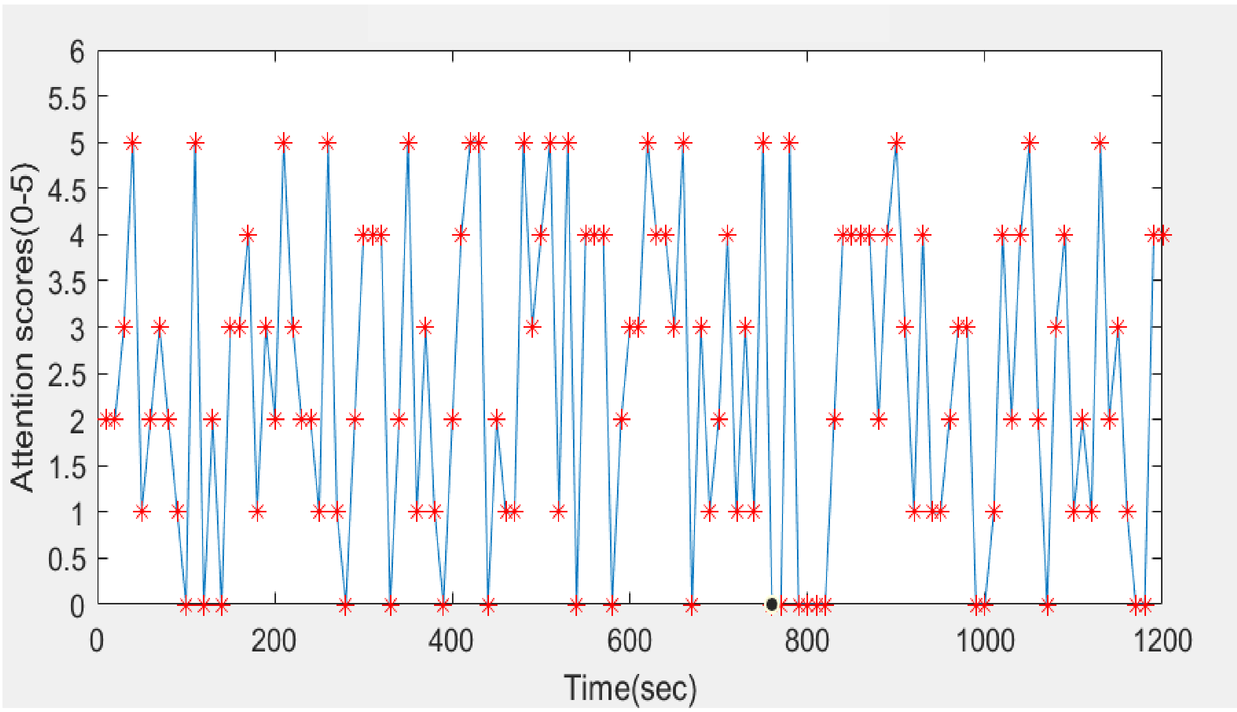

Figure 2: Attention Scores for a 20 minutes test

The MATLAB code for developing the task is given in APPENDIX A. In The two stimuli-target and non-target are created using their individual functions which are called from the main function to develop the overall structure for the game.

SECTION 2: PROCESSING THE EEG SIGNALS TO CORRELATE WITH ATTENTION OF THE PERSON

The EEG signals are collected using the EPOC+ headset developed by Emotiv company [3]. EPOC+ has 14 electrodes: AF3, F7, F3, FC5, T7, P7, O1, O2, P8, T8, FC6, F8, F4, AF4 which cover the entire circumference of the brain, uniformly distributed on both sides-right and left.

Figure 3: Emotiv EPOC+ headset [3]

EPOC+ is a non-intrusive headset which can be comfortably placed on the head. It provides easy access to professional grade brain data with quick and easy to use design [3].

EEG signal were collected from the Emotiv Headset while the person was performing the attention task. The EEG signals are collected both for training the model as well as a prediction from the model. The MATLAB code for getting the EEG signal was downloaded from Emotiv's community SDK GitHub page [4]. It allowed gathering of real time average band power signals from each of the 14 electrodes in 5 frequency bands : low-beta (1–3 Hz), theta (4–7 Hz), alpha (8–12 Hz), high-beta (13–30 Hz), and gamma (30–100 Hz).

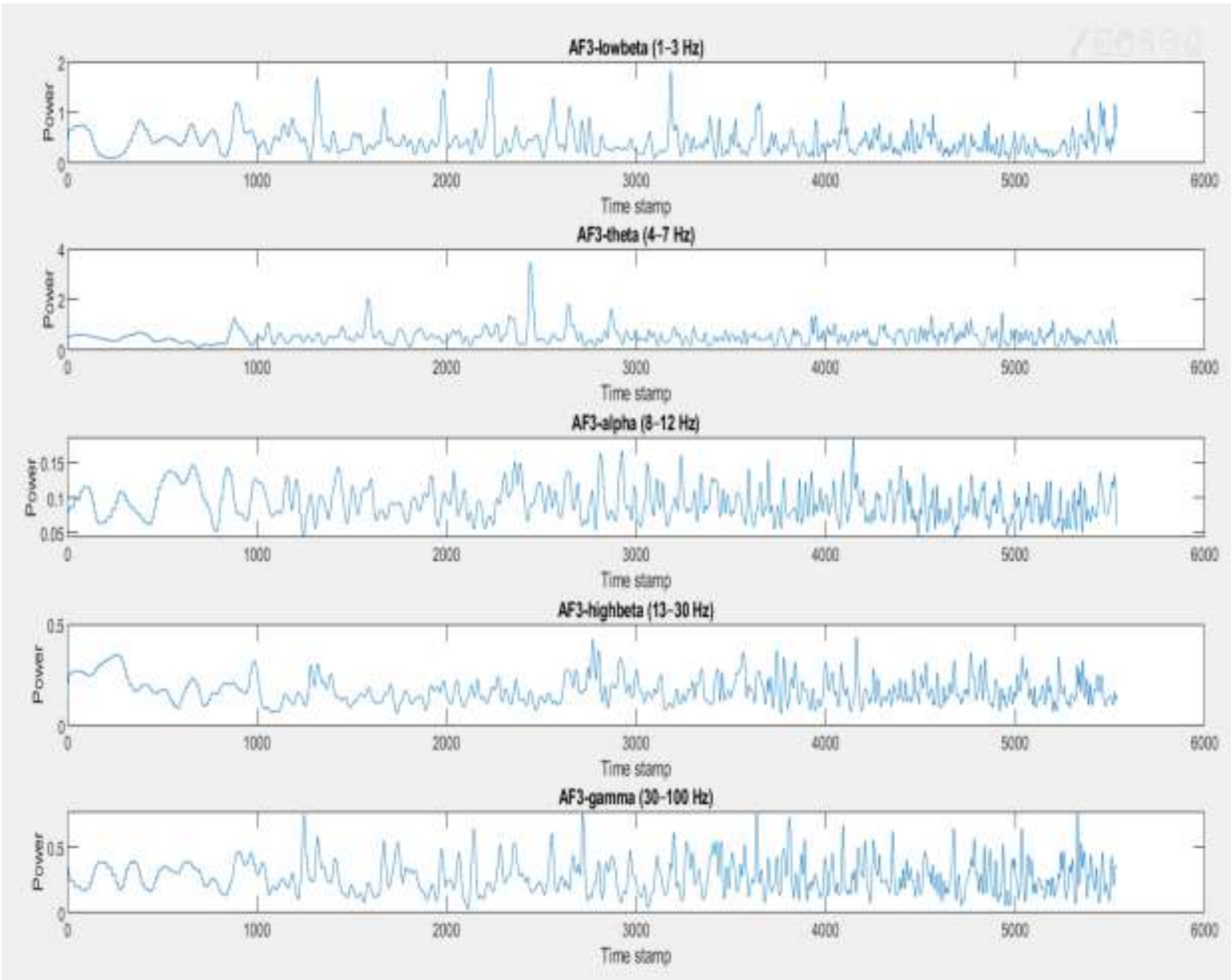

Figure 4: Electrode AF3 Power Signals in different frequency band

EEG signals were collected from the 8 electrodes: AF3, T7, P7, O1, O2, P8, T8 and AF4 and the signals from the electrodes F7, F3, FC5, FC6, F4 and F8 were discarded which are related to the motor cortex [5]. CMS and DRL electrodes are used for absolute voltage reference.

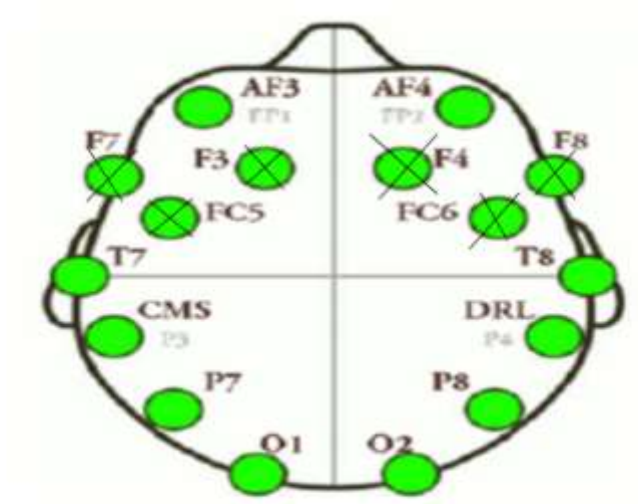

Figure 5: Discarded Electrodes

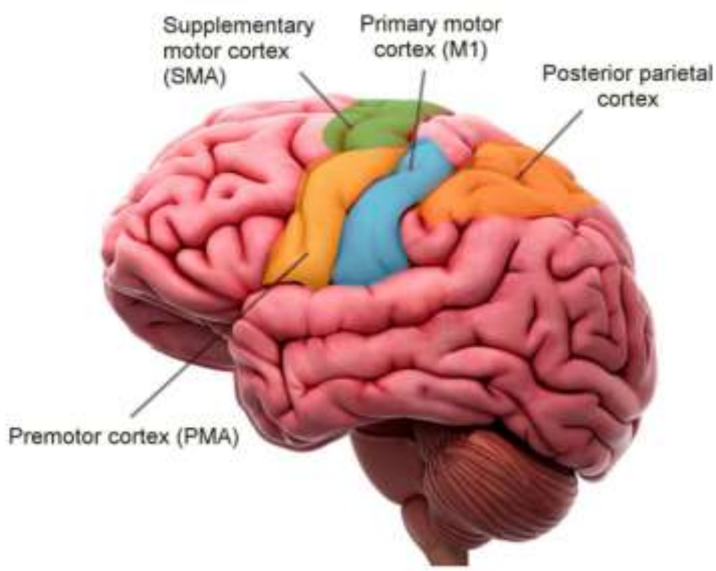

Figure 6: Principal Component of the Motor Cortex [6]

The code for processing the EEG signals is given in APPENDIX B. It was downloaded from Emotiv Github page [4]. Few changes are made in the code to store the EEG signals in proper data structures. The time for running the test can be controlled by changing the variable `‘runtime’`.

SECTION 3: CREATING A MODEL WITH THE DATA COLLECTED

Gaussian process regression model with Squared Exponential function as kernel was selected to map the EEG signals to the attention score as it generated the least RMSE (Root Mean Square Error) between the real and predicted attention scores.

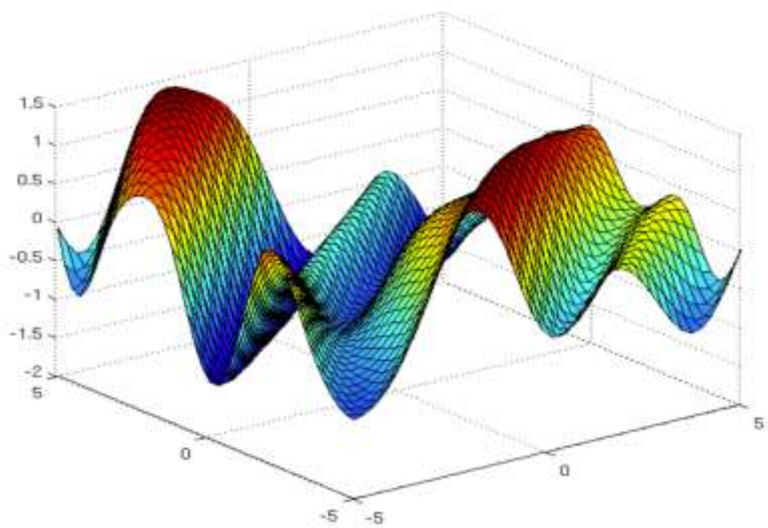

Figure 9: Multivariate Gaussian [7]

A Gaussian Process is a set of random variables such that any finite number of them have a Gaussian Distribution [8]. If $\{f(x), x \in \mathbb{R}^d, f(.): \mathbb{R}^d \rightarrow \mathbb{R}\}$ is a GP, then given $'n'$ observations, $x_1, x_2 \dots\dots x_n$, the joint distribution of the random variables $f(x_1), f(x_2) \dots\dots f(x_n)$ is a multivariate gaussian in $\mathbb{R}^n$. A Gaussian Process is defined by its mean function $\mu(x)$ where $\mu(.): \mathbb{R}^d \rightarrow \mathbb{R}$ and covariance/kernel function $k(x, \hat{x})$ where $k: \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}$. That is, if $\{f(x), x \in \mathbb{R}^d\}$ is a Gaussian Process, then $\mathbb{E}[f(x)] = \mu(x)$ and $cov(f(x), f(\hat{x})) = \mathbb{E}[\{f(x) - \mu(x)\}\{f(\hat{x}) - \mu(\hat{x})\}] = k(x, \hat{x})$

The kernel/covariance function selected for creating this Gaussian Model is Squared Exponential Kernel given by-

$k(x, \hat{x}) = \sigma_f^2 \exp\left[-\frac{1(x-\hat{x})^T(x-\hat{x})}{2\sigma_l^2}\right]$ where, σ_l −characteristic length scale

σ_f −signal standard deviation

Values of σ_l and σ_f are selected which maximizes the probability density function.

Suppose we have 'n' set of datapoints represented by $\{(x_1, y_1), (x_2, y_2) \dots \dots . (x_n, y_n)\}$ where $x \in \mathbb{R}^d$ and $y \in \mathbb{R}$. Let first $'l'$datapoints are used for training and rest $'n - l'$ are used for predicting from the model i.e., let $X = \{x_1, x_2 \dots \dots . . x_l\}$, $Y = \{y_1, y_2 \dots \dots . y_l\}$ represents the training set and $X^* = \{x_{l+1}, x_{l+2} \dots . . . x_n\}$, $Y^* = \{y_{l+1}, y_{l+2}, \dots . y_n\}$ represents the testing set. The dimensions of the sets are given as - $X \in \mathbb{R}^{l \times d}$, $X^* \in \mathbb{R}^{n-l \times d}$, $Y \in \mathbb{R}^l$ and $Y^* \in \mathbb{R}^{n-l}$

If μ and μ^* represents the mean for Y and Y^* respectively, then the multivariate Gaussian distribution is given as-

$$\begin{bmatrix} Y \\ Y^* \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mu \\ \mu^* \end{bmatrix}, \begin{bmatrix} k(X, X) & k(X, X^*) \\ k(X^*, X) & k(X^*, X^*) \end{bmatrix}\right)$$

<u>Training-</u> By marginal distribution property of multivariate gaussian, set Y itself forms a multivariate gaussian distribution s.t $Y \sim \mathcal{N}(\mu, k(X, X))$[7]. Since we know the elements in Y, we can find the mean function and parameters of the kernel function for which the logarithm of probability density function is maximum i.e., $\arg max_{\mu, \sigma_l, \sigma_f}\ p(Y)$

where, $p(Y) = \frac{\exp\left(-\frac{1}{2}(Y-\mu)^T k(X,X)(Y-\mu)\right)}{\sqrt{2\pi^l k(X,X)}}$

Testing-We have to calculate the conditional distribution of Y^* given a particular value of Y. According to the equation of conditional distribution of a multivariate gaussian [7], the distribution of Y^* is given as –

$$Y^*|Y \sim \mathcal{N}(m, D)$$

where, $m = \mu^* + k(X^*, X)k(X, X)^{-1}(Y - \mu)$

$$D = k(X^*, X^*) - k(X^*, X)k(X, X)^{-1}k(X, X^*)$$

For our experiment, EEG signals are collected while the person is performing the task for 20 minutes session. Each 20 minutes session corresponds to 120 data points consisting of the attention scores and corresponding EEG signals. Data is collected over a period of 10 days with 20 minutes sessions 4 times a day. In the end, we have 4800 data points to train our model. Out of these 4800, first 3800 attention scores were used for training and last 1000 were used for predicting the attention scores.

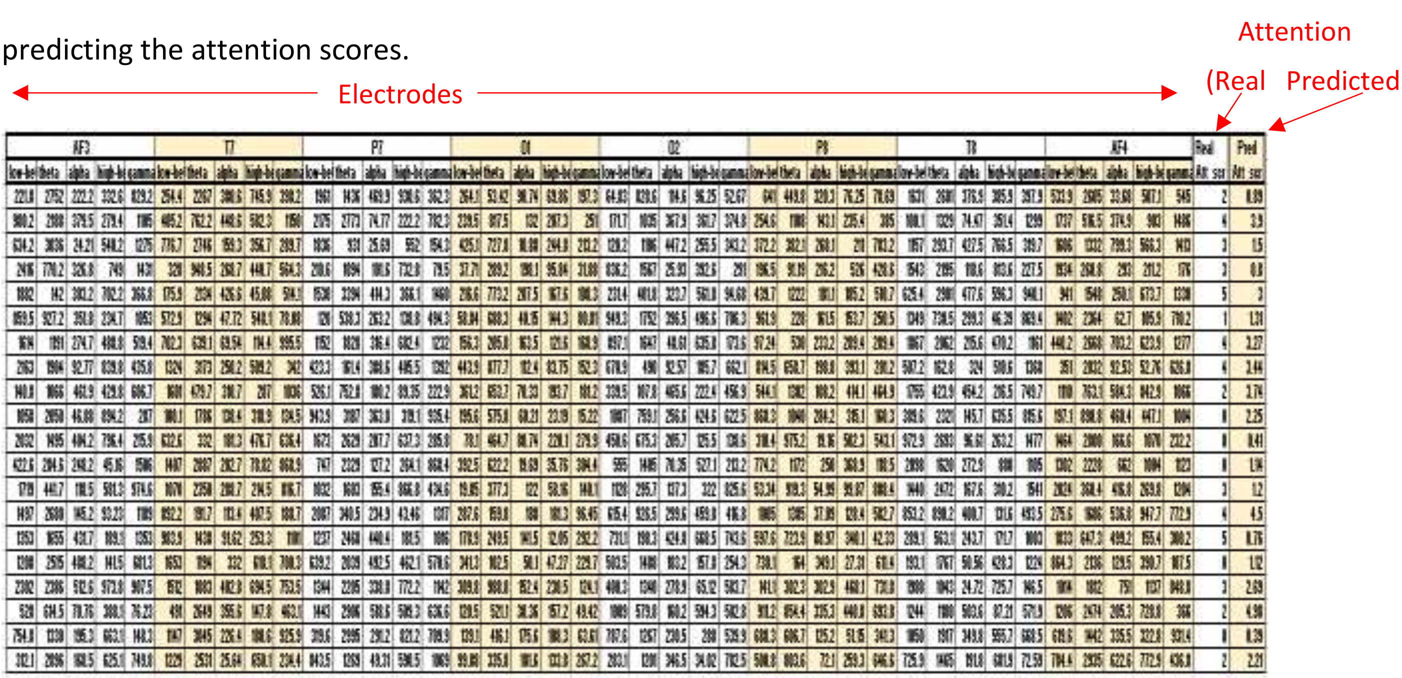

Figure 7: EEG signals vs Attention Scores

Thus, X represents average band power signals from each of the 8 electrodes in 5 frequency bands: low-beta (1–3 Hz), theta (4–7 Hz), alpha (8–12 Hz), high-beta (13–30 Hz), and gamma (30–100 Hz). So, each entry forms a vector of 40 elements that are summed over a period of 10 seconds. X forms the first 3800 EEG signals that are collected while doing the attention task, and Y represents the attention score corresponding to task.X^* represents the EEG signals for the last 1000 data and Y^* is the attention scores predicted according to the trained model.

Correlation with above equations the dimensions of the vectors are given as-

n (Total number of samples) $- 4800$

l (Number of samples for training) $- 3800$

d (Length of each training vector) $- 40$

X (Training data) $- \mathbb{R}^{3800\times 40}$

Y (Training Label) $- \mathbb{R}^{3800}$

X^* (Testing Data) $- \mathbb{R}^{1000\times 40}$

Y^* (Testing Label)$- \mathbb{R}^{1000}$

Below is a figure of predicted attention score from the model vs the attention score generated from the game. It is important to note that the attention score predicted is not exactly equal to the real scores but the rise and fall of the scores happen synchronously thus indication the change in attention level of a person from high to low and vice-versa.

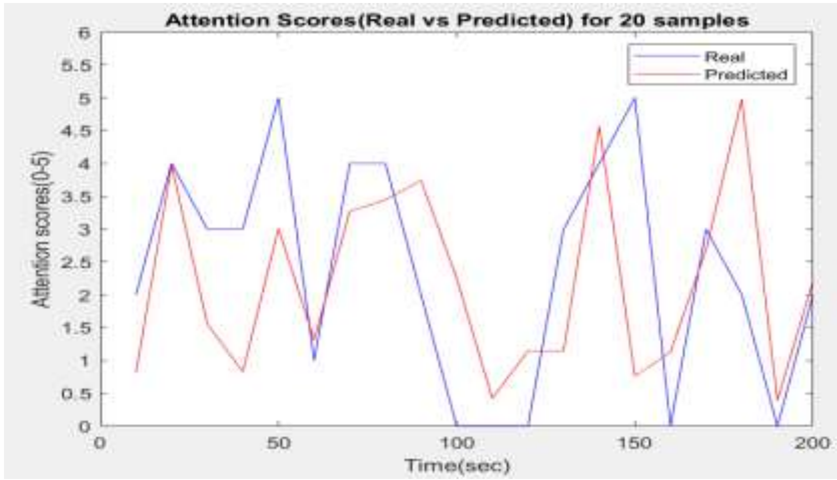

Figure 8: Attention score (Real vs Predicted)

MATALAB code for collecting the test data is given in APPENDIX C. EEG signals were mapped to appropriate labels of attention scores which are passed on to create the machine learning model. The entire dataset is used for both training and testing the model. The code for creating the Gaussian Regression Process is provided in APPENDIX D. It takes in the training dataset as input and gives the model parameters and RMSE (Root mean square error) between the predicted and real values as output.

SECTION 4: CREATING AN E-GAMING ENVIRONMENT TO INCREASE ATTENTION SPAN

The final step was to create a snake and dot game whose parameters changed according to the attention score predicted by our model. The parameters of the game which are subject to change are the speed of the game and size of the target.

The speed of the game decreases as the attention level decreases and vice-versa. Similarly, the size of target increases as attention level decreases and vice-versa. The reason for this is because it is common for people with ADHD to feel blocked when they are not able to focus. So, through this e-gaming environment, a training environment is developed where the game becomes easier when a person with ADHD is not able to focus and vice-versa, the game becomes harder when the person is attentive, to motivate them to perform better.

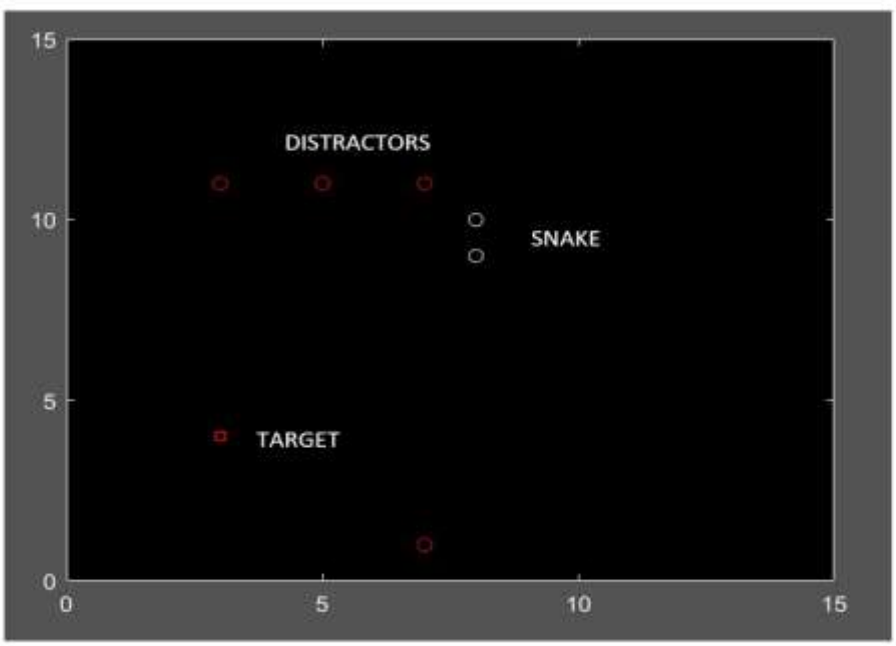

Figure 10: Attention game layout

The speed of the game and the size of the target changes every 10 seconds according to the attention score predicted by the GPR model given by the equation-

*spd=2*att_scr*

*trgtsz=10-2*att_scr*

where,

spd – speed of the game in the range of 0-10

trgtsz – target size in the range of 0-10

att_scr - attention score predicted by the GPR model in the range of 0-5

The game was tested quantitively in the lab by developing a personal training model for 5 students. The required dataset was collected over a period of 10 days while performing the attention task. A Gaussian model was developed to predict the attention level and snake game was used for training. The person playing the game was asked to concentrate on something else rather than the game and it was found that the speed of the game decreased, and the target size increased for this case. On the contrary, when the person gave his full attention to the task the parameters of the game changed oppositely. Thus, the model was able to predict the period of low attention and high attention very effectively. However, the attention scores predicted by the model were relatively similar and held the same values during the transition period from low to high attention or vice-versa.

MATLAB code for creating the snake and dot game is given in APPENDIX E. The variables 'spd' and 'trgtsz' are used to change the speed of the game and target size according to the attention of the person to regulate the difficulty level of the game.

CONCLUSION

A dynamic training environment was built which took EEG signals from Emotiv Epoc+ headset as input and predicted the attention level of a person in the range of 0-5, 0 being state of least attentiveness and 5 for maximum. Based on the state predicted, parameters of the game were changed to make the person engage in the task for higher period of time, thus aiming to increase the attention span of the person having ADHD.

The project attempts to provide a dynamic mathematical model of the brain to measure attention. All four steps can be improved individually to get a better model of the whole system. The discrete level of the attention can be increased by increasing the number of variables, apart from reaction time and key pressed. A big step forward would be to refer more neuroscience papers to have knowledge of brain maps related to attention. As advances are being made every day in trying to understand the modeling of brain function, a good machine learning algorithm to predict attention level will depend on using EEG signals from the precise lobes of the brain.

REFERENCES

(1). T.O.V.A.® 9 Clinical Manual, The TOVA Company-3321 Cerritos Avenue, Los Alamitos, CA 90720 USA, Edition Number 9.0-121-g7413481 (October 22, 2018) L5R5.

(2). Jos (10584) (2020). getkey(https://www.mathworks.com/matlabcentral/fileexchange/7465-getkey), MATLAB Central File Exchange. Retrieved March 24, 2020.

(3). https://emotiv.gitbook.io/epoc-user-manual/

(4). https://github.com/Emotiv/community-sdk

(5). Russo AA, Bittner SR, Perkins SM, et al. Motor Cortex Embeds Muscle-like Commands in an Untangled Population Response. Neuron. 2018;97(4):953–966.e8. doi:10.1016/j.neuron.2018.01.004

(6). "A MATHEMATICAL MODEL OF THE MOTOR CORTEX." (2017).

(7). C. E. Rasmussen & C. K. I. Williams, Gaussian Processes for Machine Learning, the MIT Press, 2006, ISBN 026218253X. c 2006 Massachusetts Institute of Technology. www.GaussianProcess.org/gpml

(8). https://www.mathworks.com/help/stats/gaussian-process-regression-models.html

(9). Jenson P. Current concept and controversies in the diagnosis and treatment of attention deficit hyperactivity disorder. Curr Psychiatry Rep 2000;2:102-9.

(10). https://www.mathworks.com/matlabcentral/answers/6065-creating-snake-game

APPENDIX A: MATLAB CODE TO DEVELOP ATTENTION GAME

- Main Program for the Attention Task:

```
close all;clear all;clc;

itr=10;
Scr=zeros(1,itr);
Att_scr=zeros(1,itr); %Based on number of iterations

global tar;           % Number of times Targer appears
global ScrTar;        % Entered Key and Reaction time stored here

ScrTar=zeros(100,2);
tar=0;

global nontar;         % Number of times Non-Targer appears
global ScrNonTar;      % Entered Key and Reaction time stored here
ScrNonTar=zeros(100,2);
nontar=0;

for cnt=1:itr           % Number of iterations of the game
                        % Setting up the visual screen
l=figure(6);
set(gcf,'position',[0 0 1950 1000]);

set(gcf,'Color','k')


str=tic;                 % Start of Task
stp=toc(str);

while stp<10             % Time for each task
    a=randi(2);          % Randomly selecting Target and Non-Target
    e=randi(2);
    if a==1
        Target(e);
    else
        NonTarget(e);
    end
    stp=toc(str);
end
close(l);

ScrNonTar(nontar+1:end,:)=[];
ScrTar(tar+1:end,:)=[];

cTrgCrc = 0;                %Target Correct
cNTrgCrc = 0;               %Non-Target Correct
cTrgNcrc = 0;               %Target Not Correct
cNTrgNcrc = 0;              %Non-Target Not Correct

cTrgCrcTm = 0;              %Target Correct Time
cNTrgCrcTm = 0;             %Non-Target Correct Time
cTrgNcrcTm = 0;             %Target Not Correct Time
cNTrgNcrcTm = 0;            %Non-Target Not Correct Time
```

```
for i=1:length(ScrTar')       %Calculation for Target

    if ScrTar(i,1)==32
      cTrgCrc = cTrgCrc +1;
      cTrgCrcTm = cTrgCrcTm + ScrTar(i,2);
    else
      cTrgNcrc = cTrgNcrc +1;
      cTrgNcrcTm = cTrgNcrcTm + ScrTar(i,2);
    end

end

for i=1:length(ScrNonTar')       %Calculation for Non-Target

    if ScrNonTar(i,1)==13
      cNTrgCrc = cNTrgCrc +1;
      cNTrgCrcTm = cNTrgCrcTm + ScrNonTar(i,2);
    else
      cNTrgNcrc = cNTrgNcrc +1;
      cNTrgNcrcTm = cNTrgNcrcTm + ScrNonTar(i,2);
    end

end

%Calculating Scores
Scr(1,cnt) = (cTrgCrc*cTrgCrcTm+cNTrgCrc*cNTrgCrcTm)/(cTrgCrc+cNTrgCrc+1)...
-2*(cTrgNcrc*cTrgNcrcTm+cNTrgNcrc*cNTrgNcrcTm)/(cTrgNcrc+cNTrgNcrc+1);

%Providing a discrete attention score
if Scr(1,cnt)<1
    Att_scr(1,cnt)=0;
elseif  Scr(1,cnt)>=1 && Scr(1,cnt)<1.6
    Att_scr(1,cnt)=1;
elseif Scr(1,cnt)>=1.6 && Scr(1,cnt)<2.2
    Att_scr(1,cnt)=2;
elseif Scr(1,cnt)>=2.2 && Scr(1,cnt)<2.8
    Att_scr(1,cnt)=3;
elseif Scr(1,cnt)>=2.8 && Scr(1,cnt)<3.4
    Att_scr(1,cnt)=4;
else
    Att_scr(1,cnt)=5;
end

tar=0;
nontar=0;

pause(1);
end

save('Data_att.mat','Att_scr');     %Store the attention score for training
```

- Function to design Target-

```
function[]= Target(b)
global tar;
global ScrTar;
```

```
%Target
while b>0
tar=tar+1;
l1=figure(b);
x = [4 6 6 4 4];
y = [7 7 9 9 7];
plot(x,y);
fill(x,y,'k');
axis([0 10 0 10]);
set(gca,'color','w');
set(gca,'XTick',[], 'YTick', []);
set(gcf,'position',[0 0 1950 1000]);
set(gca,'position',[0.45 0.6 0.15 0.15]);
set(gca,'DataAspectRatio',[1 1 1]);
set(gcf,'Color','k');

pause(0.01);            % Time for display
close(l1);
b=b-1;

[Q,T1] = getkey(1);
ScrTar(tar,1)=Q;
ScrTar(tar,2)=T1;
end
end
```

- Function to design Non-Target-

```
function[]=NonTarget(b)
global nontar;
global ScrNonTar;
%NonTarget
while b>0
nontar=nontar+1;
l2=figure(b);
x = [4 6 6 4 4];
y = [1 1 3 3 1];
plot(x,y);
fill(x,y,'k');
axis([0 10 0 10]);
set(gca,'color','w');
set(gca,'XTick',[], 'YTick', []);
set(gcf,'position',[0 0 1950 1000]);
set(gca,'position',[0.45 0.6 0.15 0.15]);
set(gca,'DataAspectRatio',[1 1 1]);
set(gcf,'Color','k');
pause(0.01);
close(l2);
b=b-1;
[P,T2] = getkey(1);
ScrNonTar(nontar,1)=P;
ScrNonTar(nontar,2)=T2;
end
end
```

APPENDIX B: MATLAB CODE FOR PROCESSING EEG SIGNALS [4]-

- Program for getting EEG signals [4]-

```
w = warning ('off','all');
bitVersion = computer('arch');
if (strcmp(bitVersion, 'win64'))
    loadlibrary('D:/Spring_Courses_2020/Project/community-sdk-
master/community-sdk-
master/bin/win64/edk.dll','D:/Spring_Courses_2020/Project/community-sdk-
master/community-sdk-
master/include/Iedk.h','addheader','IedkErrorCode.h','addheader','IEmoStateDL
L.h','addheader','FacialExpressionDetection.h','addheader','MentalCommandDete
ction.h','addheader','IEmotivProfile.h','addheader','EmotivLicense.h','alias'
,'libIEDK');
    loadlibrary('D:/Spring_Courses_2020/Project/community-sdk-
master/community-sdk-
master/bin/win64/edk.dll','D:/Spring_Courses_2020/Project/community-sdk-
master/community-sdk-master/include/IEmoStateDLL.h', 'alias',
'libIEmoStateDLL');
else
    loadlibrary('C:/Users/deyag/Downloads/community-sdk-master/community-sdk-
master/bin/win32/edk.dll','C:/Users/deyag/Downloads/community-sdk-
master/community-sdk-
master/include/Iedk.h','addheader','IedkErrorCode.h','addheader','IEmoStateDL
L.h','addheader','FacialExpressionDetection.h','addheader','MentalCommandDete
ction.h','addheader','IEmotivProfile.h','addheader','EmotivLicense.h','alias'
,'libIEDK');
    loadlibrary('C:/Users/deyag/Downloads/community-sdk-master/community-sdk-
master/bin/win32/edk.dll','C:/Users/deyag/Downloads/community-sdk-
master/community-sdk-master/include/IEmoStateDLL.h',
'alias','libIEmoStateDLL');
end

% libfunctionsview('edk');
EDK_OK = 0;

%Full enum channels:
%Hard-coded enum value based on IEE_DataChannels_enum (Iedk.h) for Insight
headset sensor:
dataChannel = struct('IED_AF3', 3, 'IED_F7',4, 'IED_F3', 5, 'IED_FC5', 6,
'IED_T7', 7,'IED_P7', 8, 'IED_Pz', 9,'IED_O2', 10, 'IED_P8', 11, 'IED_T8',
12, 'IED_FC6', 13, 'IED_F4', 14, 'IED_F8', 15, 'IED_AF4', 16);

channelName = {'IED_AF3', 'IED_F7', 'IED_F3', 'IED_FC5', 'IED_T7','IED_P7',
'IED_Pz','IED_O2', 'IED_P8', 'IED_T8', 'IED_FC6', 'IED_F4',
'IED_F8','IED_AF4'};
res = calllib('libIEDK','IEE_EngineConnect', 'Emotiv Systems-5');
eEvent = calllib('libIEDK','IEE_EmoEngineEventCreate');
eState = calllib('libIEDK','IEE_EmoStateCreate');

% run 20 minutes
runtime = 1200;
fprintf('Run time: %d \n', runtime);
userAdded = false;
numberSamplePtr = libpointer('uint32Ptr', 0);
thetaPtr = libpointer('doublePtr', 0);
```

```
alphaPtr = libpointer('doublePtr', 0);
lowBetaPtr = libpointer('doublePtr', 0);
highBetaPtr = libpointer('doublePtr', 0);
gammaPtr = libpointer('doublePtr', 0);
userIdPtr = libpointer('uint32Ptr', 0);
tic;

while (toc < runtime)
    state = calllib('libIEDK','IEE_EngineGetNextEvent',eEvent);

    if(state == EDK_OK)
        eventType = calllib('libIEDK','IEE_EmoEngineEventGetType',eEvent);
        calllib('libIEDK','IEE_EmoEngineEventGetUserId',eEvent, userIdPtr);
        if (strcmp(eventType,'IEE_UserAdded') == true)
            fprintf('User added: %d', userIdPtr.Value)
            userAdded = true;
        end
    end

    if (userAdded)
        if strcmp(eventType,'IEE_EmoStateUpdated') == true
            thetaPtr.Value = 0;
            alphaPtr.Value = 0;
            lowBetaPtr.Value = 0;
            highBetaPtr.Value = 0;
            gammaPtr.Value = 0;
            for index = 1 : numel(channelName)
                res = calllib('libIEDK','IEE_GetAverageBandPowers',
userIdPtr.Value, dataChannel.([channelName{index}]), thetaPtr, alphaPtr,
lowBetaPtr, highBetaPtr, gammaPtr);
                if (res == EDK_OK)
                    fprintf('theta: %f , alpha: %f , low beta: %f , high
beta: %f , gamma: %f , channel: %s \n', thetaPtr.Value, alphaPtr.Value,
lowBetaPtr.Value, highBetaPtr.Value, gammaPtr.Value, channelName{index});
                    theta1=[theta1; thetaPtr.Value];
                    alpha1=[alpha1; alphaPtr.Value];
                    lowbeta1=[lowbeta1; lowBetaPtr.Value];
                    highbeta1=[ highbeta1; highBetaPtr.Value];
                    gamma1=[gamma1;gammaPtr.Value];
                    i=length( theta1);
                     if  rem(i,14)==1
                         AF3theta=[AF3theta;thetaPtr.Value ];
                         AF3alpha=[AF3alpha;alphaPtr.Value ];
                         AF3gamma=[AF3gamma;gammaPtr.Value ];
                         AF3highbeta=[AF3highbeta;highBetaPtr.Value ];
                         AF3lowbeta=[AF3lowbeta;lowBetaPtr.Value ];
                     end
                     if  rem(i,14)==2
                         F7theta=[F7theta;thetaPtr.Value ];
                         F7alpha=[F7alpha;alphaPtr.Value ];
                         F7gamma=[F7gamma;gammaPtr.Value ];
                         F7highbeta=[F7highbeta;highBetaPtr.Value ];
                         F7lowbeta=[F7lowbeta;lowBetaPtr.Value ];
                     end
                      if  rem(i,14)==3
                         F3theta=[F3theta;thetaPtr.Value ];
                         F3alpha=[F3alpha;alphaPtr.Value ];
```

```
    F3gamma=[F3gamma;gammaPtr.Value ];
    F3highbeta=[F3highbeta;highBetaPtr.Value ];
    F3lowbeta=[F3lowbeta;lowBetaPtr.Value ];
 end
if  rem(i,14)==4
    FC5theta=[FC5theta;thetaPtr.Value ];
    FC5alpha=[FC5alpha;alphaPtr.Value ];
    FC5gamma=[FC5gamma;gammaPtr.Value ];
    FC5highbeta=[FC5highbeta;highBetaPtr.Value ];
    FC5lowbeta=[FC5lowbeta;lowBetaPtr.Value ];
end
 if  rem(i,14)==5
    T7theta=[T7theta;thetaPtr.Value ];
    T7alpha=[T7alpha;alphaPtr.Value ];
    T7gamma=[T7gamma;gammaPtr.Value ];
    T7highbeta=[T7highbeta;highBetaPtr.Value ];
    T7lowbeta=[T7lowbeta;lowBetaPtr.Value ];
 end
  if  rem(i,14)==6
    P7theta=[P7theta;thetaPtr.Value ];
    P7alpha=[P7alpha;alphaPtr.Value ];
    P7gamma=[P7gamma;gammaPtr.Value ];
    P7highbeta=[P7highbeta;highBetaPtr.Value ];
    P7lowbeta=[P7lowbeta;lowBetaPtr.Value ];
  end
  if  rem(i,14)==7
    Pztheta=[ Pztheta;thetaPtr.Value ];
    Pzalpha=[ Pzalpha;alphaPtr.Value ];
    Pzgamma=[ Pzgamma;gammaPtr.Value ];
    Pzhighbeta=[ Pzhighbeta;highBetaPtr.Value ];
    Pzlowbeta=[ Pzlowbeta;lowBetaPtr.Value ];
end
if  rem(i,14)==8
    IED_O2theta=[ IED_O2theta;thetaPtr.Value ];
    IED_O2alpha=[ IED_O2alpha;alphaPtr.Value ];
    IED_O2gamma=[ IED_O2gamma;gammaPtr.Value ];
    IED_O2highbeta=[IED_O2highbeta;highBetaPtr.Value ];
    IED_O2lowbeta=[ IED_O2lowbeta;lowBetaPtr.Value ];
end
 if  rem(i,14)==9
   P8theta=[ P8theta;thetaPtr.Value ];
   P8alpha=[ P8alpha;alphaPtr.Value ];
   P8gamma=[ P8gamma;gammaPtr.Value ];
   P8highbeta=[P8highbeta;highBetaPtr.Value ];
   P8lowbeta=[ P8lowbeta;lowBetaPtr.Value ];
 end
 if  rem(i,14)==10
   IED_T8theta=[IED_T8theta;thetaPtr.Value ];
   IED_T8alpha=[ IED_T8alpha;alphaPtr.Value ];
   IED_T8gamma=[ IED_T8gamma;gammaPtr.Value ];
   IED_T8highbeta=[IED_T8highbeta;highBetaPtr.Value ];
   IED_T8lowbeta=[ IED_T8lowbeta;lowBetaPtr.Value ];
 end
 if  rem(i,14)==11
   IED_FC6theta=[IED_FC6theta;thetaPtr.Value ];
   IED_FC6alpha=[ IED_FC6alpha;alphaPtr.Value ];
   IED_FC6gamma=[ IED_FC6gamma;gammaPtr.Value ];
```

```
                        IED_FC6highbeta=[IED_FC6highbeta;highBetaPtr.Value ];
                        IED_FC6lowbeta=[ IED_FC6lowbeta;lowBetaPtr.Value ];
                      end
                      if  rem(i,14)==12
                        IED_F4theta=[IED_F4theta;thetaPtr.Value ];
                        IED_F4alpha=[ IED_F4alpha;alphaPtr.Value ];
                        IED_F4gamma=[ IED_F4gamma;gammaPtr.Value ];
                        IED_F4highbeta=[IED_F4highbeta;highBetaPtr.Value ];
                        IED_F4lowbeta=[ IED_F4lowbeta;lowBetaPtr.Value ];
                       end
                       if  rem(i,14)==13
                        IED_F8theta=[ IED_F8theta;thetaPtr.Value ];
                        IED_F8alpha=[  IED_F8alpha;alphaPtr.Value ];
                        IED_F8gamma=[  IED_F8gamma;gammaPtr.Value ];
                        IED_F8highbeta=[ IED_F8highbeta;highBetaPtr.Value ];
                        IED_F8lowbeta=[  IED_F8lowbeta;lowBetaPtr.Value ];
                       end
                       if  rem(i,14)==0
                        IED_AF4theta=[ IED_AF4theta;thetaPtr.Value ];
                        IED_AF4alpha=[  IED_AF4alpha;alphaPtr.Value ];
                        IED_AF4gamma=[  IED_AF4gamma;gammaPtr.Value ];
                        IED_AF4highbeta=[ IED_AF4highbeta;highBetaPtr.Value
];

                        IED_AF4lowbeta=[  IED_AF4lowbeta;lowBetaPtr.Value ];
                       end


                        AF3=[AF3alpha AF3theta AF3gamma  AF3highbeta
AF3lowbeta];
                        F7=[ F7alpha F7theta  F7gamma F7highbeta F7lowbeta];
                        F3=[ F3alpha F3theta  F3gamma F3highbeta F3lowbeta];
                        FC5=[ FC5alpha FC5theta FC5gamma FC5highbeta
FC5lowbeta];
                        T7=[  T7alpha  T7theta  T7gamma T7highbeta
T7lowbeta];
                        P7=[ P7alpha  P7theta P7gamma P7highbeta P7lowbeta];
                        Pz=[Pzalpha Pztheta Pzgamma  Pzhighbeta Pzlowbeta];
                        IED_O2=[IED_O2alpha  IED_O2theta IED_O2gamma
IED_O2highbeta IED_O2lowbeta];
                        P8=[ P8alpha P8theta P8gamma P8highbeta P8lowbeta];
                        IED_T8=[ IED_T8alpha IED_T8theta  IED_T8gamma
IED_T8highbeta IED_T8lowbeta];
                        IED_FC6=[IED_FC6alpha IED_FC6theta  IED_FC6gamma
IED_FC6highbeta IED_FC6lowbeta];
                        IED_F4=[IED_F4alpha IED_F4theta IED_F4gamma
IED_F4highbeta IED_F4lowbeta];
                        IED_F8=[IED_F8alpha IED_F8theta IED_F8gamma
IED_F8highbeta IED_F8lowbeta];
                        IED_AF4=[IED_AF4alpha IED_AF4theta IED_AF4gamma
IED_AF4highbeta IED_AF4lowbeta];

                       %Discarding the Excess Value per Iteration without
Overwritting

                        I=size(IED_AF4,1)+1;
                        if I>14
                            AF3new=AF3;
```

```
                                        AF3new(I:end,:)=[];
                                        T7new=T7;
                                        T7new(I:end,:)=[];
                                        P7new=P7;
                                        P7new(I:end,:)=[];
                                        Pznew=Pz;
                                        Pznew(I:end,:)=[];
                                        IED_O2new=IED_O2;
                                        IED_O2new(I:end,:)=[];
                                        P8new=P8;
                                        P8new(I:end,:)=[];
                                        IED_T8new=IED_T8;
                                        IED_T8new(I:end,:)=[];
                                        IED_AF4new=IED_AF4;
                                        IED_AF4new(I:end,:)=[];
                                        Data=[AF3new T7new P7new Pznew IED_O2new P8new
IED_T8new IED_AF4new];

                                    end
                                end
                            end
                        end
                    end
end
calllib('libIEDK','IEE_EngineDisconnect')
calllib('libIEDK','IEE_EmoStateFree',eState);
calllib('libIEDK','IEE_EmoEngineEventFree',eEvent);
save('Data_eeg.mat','Data1');
disp('finish');
%end
```

APPENDIX C: MATLAB CODE TO COLLECT TEST DATA-

- Main Code for collecting the Test Data-

```
clear all;
load('Data_eeg.mat');
load('Data_att.mat');

cnt = round(size(Data1,1)/size(Att_scr,2))

TstDt=zeros(size(Att_scr,2),41);
TstDt(:,41)=Att_scr';

for i=1:size(Att_scr,2)
    if(i*cnt<size(Att_scr,2))
        TstDt(i,1:40)=sum(Data1((i-1)*cnt+1:i*cnt,:));
    else
        TstDt(i,1:40)=sum(Data1((i-1)*cnt+1:end,:));
    end
end
%Model-
[trainedModel, validationRMSE] = trainRegressionModel(TstDt(1:3800,:));
% Predicted attention score
 Att_scr_pred=trainedModel.predictFcn(TstDt(3801:end,1:40));

save('TrnData.mat','trainedModel','cnt')
```

APPENDIX D: MATLAB CODE TO CREATE GAUSSIAN PROCESS REGRESSION MODEL WITH SQUARED EXPONENTIAL FUNCTION AS KERNEL [8]-

```
function [trainedModel, validationRMSE] = trainRegressionModel(trainingData)
% [trainedModel, validationRMSE] = trainRegressionModel(trainingData)
% returns a trained regression model and its RMSE. This code recreates the
% model trained in Regression Learner app. Use the generated code to
% automate training the same model with new data, or to learn how to
% programmatically train models.
%
%  Input:
%      trainingData: a matrix with the same number of columns and data type
%       as imported into the app.
%
%  Output:
%      trainedModel: a struct containing the trained regression model. The
%       struct contains various fields with information about the trained
%       model.
%
%      trainedModel.predictFcn: a function to make predictions on new data.
%
%      validationRMSE: a double containing the RMSE. In the app, the
%       History list displays the RMSE for each model.
%
% Use the code to train the model with new data. To retrain your model,
% call the function from the command line with your original data or new
% data as the input argument trainingData.
%
% For example, to retrain a regression model trained with the original data
% set T, enter:
%   [trainedModel, validationRMSE] = trainRegressionModel(T)
%
% To make predictions with the returned 'trainedModel' on new data T2, use
%   yfit = trainedModel.predictFcn(T2)
%
% T2 must be a matrix containing only the predictor columns used for
% training. For details, enter:
%   trainedModel.HowToPredict

% Auto-generated by MATLAB on 20-Mar-2020 21:10:19


% Extract predictors and response
% This code processes the data into the right shape for training the
% model.
% Convert input to table
inputTable = array2table(trainingData, 'VariableNames', {'column_1',
'column_2', 'column_3', 'column_4', 'column_5', 'column_6', 'column_7',
'column_8', 'column_9', 'column_10', 'column_11', 'column_12', 'column_13',
'column_14', 'column_15', 'column_16', 'column_17', 'column_18', 'column_19',
'column_20', 'column_21', 'column_22', 'column_23', 'column_24', 'column_25',
'column_26', 'column_27', 'column_28', 'column_29', 'column_30', 'column_31',
```

```
'column_32', 'column_33', 'column_34', 'column_35', 'column_36', 'column_37',
'column_38', 'column_39', 'column_40', 'column_41'});

predictorNames = {'column_1', 'column_2', 'column_3', 'column_4', 'column_5',
'column_6', 'column_7', 'column_8', 'column_9', 'column_10', 'column_11',
'column_12', 'column_13', 'column_14', 'column_15', 'column_16', 'column_17',
'column_18', 'column_19', 'column_20', 'column_21', 'column_22', 'column_23',
'column_24', 'column_25', 'column_26', 'column_27', 'column_28', 'column_29',
'column_30', 'column_31', 'column_32', 'column_33', 'column_34', 'column_35',
'column_36', 'column_37', 'column_38', 'column_39', 'column_40'};
predictors = inputTable(:, predictorNames);
response = inputTable.column_41;
isCategoricalPredictor = [false, false, false, false, false, false, false,
false, false, false, false, false, false, false, false, false, false, false,
false, false, false, false, false, false, false, false, false, false, false,
false, false, false, false, false, false, false, false, false, false, false];

% Train a regression model
% This code specifies all the model options and trains the model.
regressionGP = fitrgp(...
    predictors, ...
    response, ...
    'BasisFunction', 'constant', ...
    'KernelFunction', 'exponential', ...
    'Standardize', true);

% Create the result struct with predict function
predictorExtractionFcn = @(x) array2table(x, 'VariableNames',
predictorNames);
gpPredictFcn = @(x) predict(regressionGP, x);
trainedModel.predictFcn = @(x) gpPredictFcn(predictorExtractionFcn(x));

% Add additional fields to the result struct
trainedModel.RegressionGP = regressionGP;
trainedModel.About = 'This struct is a trained model exported from Regression
Learner R2018b.';
trainedModel.HowToPredict = sprintf('To make predictions on a new predictor
column matrix, X, use: \n  yfit = c.predictFcn(X) \nreplacing ''c'' with the
name of the variable that is this struct, e.g. ''trainedModel''. \n \nX must
contain exactly 40 columns because this model was trained using 40
predictors. \nX must contain only predictor columns in exactly the same order
and format as your training \ndata. Do not include the response column or any
columns you did not import into the app. \n \nFor more information, see <a
href="matlab:helpview(fullfile(docroot, ''stats'', ''stats.map''),
''appregression_exportmodeltoworkspace'')">How to predict using an exported
model</a>.');

% Extract predictors and response
% This code processes the data into the right shape for training the
% model.
% Convert input to table
inputTable = array2table(trainingData, 'VariableNames', {'column_1',
'column_2', 'column_3', 'column_4', 'column_5', 'column_6', 'column_7',
'column_8', 'column_9', 'column_10', 'column_11', 'column_12', 'column_13',
'column_14', 'column_15', 'column_16', 'column_17', 'column_18', 'column_19',
'column_20', 'column_21', 'column_22', 'column_23', 'column_24', 'column_25',
'column_26', 'column_27', 'column_28', 'column_29', 'column_30', 'column_31',
```

```
'column_32', 'column_33', 'column_34', 'column_35', 'column_36', 'column_37',
'column_38', 'column_39', 'column_40', 'column_41'});

predictorNames = {'column_1', 'column_2', 'column_3', 'column_4', 'column_5',
'column_6', 'column_7', 'column_8', 'column_9', 'column_10', 'column_11',
'column_12', 'column_13', 'column_14', 'column_15', 'column_16', 'column_17',
'column_18', 'column_19', 'column_20', 'column_21', 'column_22', 'column_23',
'column_24', 'column_25', 'column_26', 'column_27', 'column_28', 'column_29',
'column_30', 'column_31', 'column_32', 'column_33', 'column_34', 'column_35',
'column_36', 'column_37', 'column_38', 'column_39', 'column_40'};
predictors = inputTable(:, predictorNames);
response = inputTable.column_41;
isCategoricalPredictor = [false, false, false, false, false, false, false,
false, false, false, false, false, false, false, false, false, false, false,
false, false, false, false, false, false, false, false, false, false, false,
false, false, false, false, false, false, false, false, false, false, false];

% Perform cross-validation
partitionedModel = crossval(trainedModel.RegressionGP, 'KFold', 5);

% Compute validation predictions
validationPredictions = kfoldPredict(partitionedModel);

% Compute validation RMSE
validationRMSE = sqrt(kfoldLoss(partitionedModel, 'LossFun', 'mse'));
```

APPENDIX E: MATLAB CODE TO DEVELOP SNAKE AND DOT GAME [10]-

```
function snake_game_changes(att_scr)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%     PRESS 'Q' TO EXIT GAME
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%close all
%OPTIONS%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
spd=2*att_scr;             %difficulty:  1-10
bounds=1;                 %bounds? 1-yes 0-no
trgtsz=10-2*att_scr;
player_name = 'Agnibh';
axis_limit= 15;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
d=0; %direction
%Tracking starting coordinates and direction
x =round(axis_limit/2); %starting point
y =round(axis_limit/2); %starting point
d =randi([1,4]);% generates random direction to start in for snake
%Target starting coordinates
a =randi([1 axis_limit-1],1);%generates random x coordinate for food
b =randi([1 axis_limit-1],1);%generates random y coordinate for food
snake(1,1:2)=[x y];%defines the snake for x and y coordinates
size_snake=1;
ate=1;   %snake ate food
ex=0;    % used to exit game
food=[a b];%defines food for a and b coordinates
draw_snake(snake,food,size_snake,axis_limit,trgtsz);
figure('KeyPressFcn',@my_callback);
    function my_callback(~,event)%callback function for movement
        switch event.Character
            case 'q'
                ex=1;
            case 30               % arrow direction
                if(d~=2)
                    d = 1;              %up d=1
                end
            case 31
                if(d~=1)
                    d = 2;              %down d=2
                end
            case 29
                if(d~=4)
                    d = 3;              %right d=3
                end
            case 28
                if(d~=3)
                    d = 4;              %left d=4
                end
        end
    end
while (ex~=1)%runs the snake as long as q is not pressed
    size_snake=size(snake);
    size_snake=size_snake(1);
    for l=size_snake+ate:-1:2
        snake(l,:)=snake(l-1,:);
```

```
    end
    switch d          %calling callback function
        case 1
            snake(1,2)=snake(1,2)+1;%add value of 1 to y position
        case 2
            snake(1,2)=snake(1,2)-1;%subtract value of 1 to y position
        case 3
            snake(1,1)=snake(1,1)+1;%add value of 1 to x position
        case 4
            snake(1,1)=snake(1,1)-1;%subtracts value of 1 to x position
    end
    draw_snake(snake,food,size_snake,axis_limit,trgtsz);%draws the snake
     %draws the snake & distracters
      pause(max([(105-spd*10)/(10*axis_limit) .001])) %diffculty makes game
faster;
      if snake(1,1)==food(1) && snake(1,2)==food(2)%if the snake and food are
in the same position
          ate=1;
          food(1) = randi([1 axis_limit-1]);%creates a new x position for the
food
          food(2) = randi([1 axis_limit-1]);%creates a new y position for the
food
      else
          ate=0;
      end
      if bounds==1
          if snake(1,1)==0 %if snake exceeds boundaries display message box
              msgbox({player_name;'Stay Determined!'},'Game Over')
              ex=1;
          elseif snake(1,2)==0%if snake exceeds boundaries display message
box
              msgbox({player_name;'Stay Determined!'},'Game Over')
              ex=1;
          elseif snake(1,1)==axis_limit%if snake exceeds boundaries display
message box
              msgbox({player_name;'Stay Determined!'},'Game Over')
              ex=1;
          elseif snake(1,2)==axis_limit%if snake exceeds boundaries display
message box
              msgbox({player_name;'Stay Determined!'},'Game Over')
              ex=1;
          end
      else
          snake=snake-((snake>axis_limit).*(axis_limit+1));
          snake=snake+((snake<0).*(axis_limit+1));
      end
      if (sum(snake(:, 1) ==snake(1, 1)   & snake(:, 2) == snake(1, 2)
)>1)%if snake hits itself
          msgbox({player_name;'Stay Determined!'},'Game Over')
          break
      end
  end
  close all
  end
function draw_snake(snake,food,size_snake,axis_limit,trgtsz)
      for p = 1:size_snake
          plot(snake(p,1),snake(p,2), 'wo')
```

```
            hold on
        end
        plot(food(1,1),food(1,2), 'rs','LineWidth', trgtsz)%creates the vectors
for the food and snake and plots them
        whitebg([0 0 0])%creates black background
        axis([0, axis_limit, 0, axis_limit])%creates the axis for gameplay
        hold on

        distracter_generator(axis_limit); % see function
        hold off
end
function distracter_generator(axis_limit)
    %Generates white distracters every frame
    j = randi([1 axis_limit-1],1); %x random
    k = randi([1 axis_limit-1],1); %y random
    l = randi([1 axis_limit-1],1); %x random
    m = randi([1 axis_limit-1],1); %y random
    n = randi([1 axis_limit-1],1); %x random
    o = randi([1 axis_limit-1],1); %y random
    q = randi([1 axis_limit-1],1); %x random
    r = randi([1 axis_limit-1],1); %y random

    %Coordinates of distracters
    plot(j,k, 'ro')
    %Plots them as white circles.
    plot(l,m, 'ro')
    %Plots red triangles
    plot(n,o, 'ro')
    %Plots them as white circles.
    plot(q,r, 'ro')
    %Plots red triangles
end
```