Training a Gaming Agent on Brainwaves

Bartolomé Francisco, Moreno Juan, Navas Natalia, Vitali José,
Rodrigo Ramele, *Member, IEEE,* Juan Miguel, Santos

***Abstract*—The field of Brain Computer Interfaces has seen a bloom in the past few years. One of its applications is the training of systems using signals originated from the human brain. Out of all the methods used to acquire information from the central nervous system the one that has gained the most popularity is Electroencephalography (EEG), mainly because of its effectiveness, low cost and because it is noninvasive. Event-related potential (ERP) are brain signals that are time-locked to a particular event. Error-related potential (ErrP) are a particular type of ERP that are can be elicited by a person who attends a recognizable error. When subjects observe an agent who makes a mistake this potential can be discerned when analyzing the signals which are directly related with the cognitive interpretation of an erroneous outcome or decision. Reinforcement Learning (RL) is an approach to machine learning where an agent tries to maximize the reward obtained while interacting with an unknown environment. This work follow this path by detecting the ErrP potential from subjects witnessing an agent playing the game and improving the agent behavior using RL and obtaining rewards from the subjects.**

***Index Terms*—ErrP, BCI, EEG, RL, Agent, AI**

I. Introduction

THE effectiveness of today's human–machine interaction and artificial intelligence is limited by a communication bottleneck as humans are required to translate high-level concepts into a machine-mandated sequence of instructions [1]. This work tackles this problem by exploring the use of brain signals as an interface between human and computer, trying to overcome this limitation by providing information to the system without explicit communication from the user. This information is used to make a gaming agent improve its operational performance using electroencephalography (EEG) signals as feedback of the performed task from an observational human critic. The idea is that subjects observing a computer game play can train the gaming agent using only signals components called Error-related Potentials (ErrP) that can be identified from their brain signals. These can be found on EEG traces and are elicited when a subject is aware of the presence of un unexpected outcome, which she identifies as an error. It is currently an extensive area of research in the neuroscience community [2]. They can detected by signal processing and machine learning techniques [3] and they are also used in Brain-Computer Interfaces (BCI) to implement or enhance artificial communication channels [4].

In order to train the agent, a natural solution for this scenario comes handy from the field of Artificial Intelligence: Reinforcement Learning (RL). Recently, this technique has seen a come-back. Nonneglected is the influence of DeepBrain's AlphaGo project, which was the first to reach superhuman proficiency when it won the complex game Go against several world champions [5].

Previous research has explored the usage of RL with reward signals based on brain activity, recorded by an EEG-based BCI system during the task execution. The papers [6], [7] have successfully demonstrated that a robot can be controlled by obtaining a reward signal from brain activity of a person which is observing the robot solving a task. Other approaches has used the signal as an important feedback to identify targets for robots when implementing shared-control strategies [8]. Additionally, ErrPs has been used in the context of games as an additional feedback channel that can be explored to enhance gaming experience [9]. In this same line, the objective of this work is to use the information extracted from brainwaves to enhance the performance of a gaming agent, replicating and extending those results.

In Section II the general layout of the cognitive game experiment is described. Section III outlines the processing pipeline used to detect ErrP components. The following Section **??** describes the classification algorithm and their parameters. Results and conclusions are exposed at the end.

II. Materials and Methods

This experiment consists of collecting signals from a person's brain while they are watching a game where the agent knows the game rules but not how to win it. Hence, the agent can learn the winning strategy using the person's feedback to improve its own performance. This section describes the process of obtaining the person's brain signals and how they are used to make the agent more efficient.

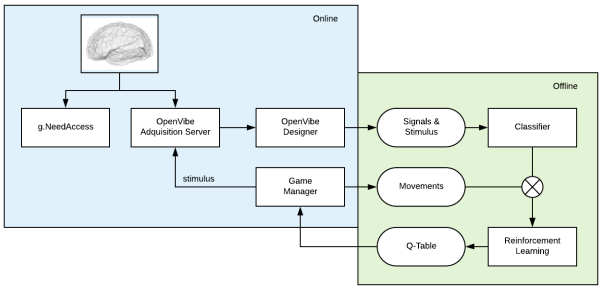


Fig. 1: Overview of the experimental procedure.

The experimental procedure is summarized in Figure 1. The core of the procedure is the retrieval of the subject's brain activity. This process is called brainwave session. Subjects are recruited voluntarily. They are given a consent form with

R. Ramele and J.M.Santos are with the Department of Computer Engineering, Instituto Tecnológico de Buenos Aires(ITBA), Argentina, e-mail: rramele@itba.edu.ar.

Manuscript received April 19, 2005; revised August 26, 2015.

questions regarding their health (previous health issues and particular visual sensitivity), habits (sleeping hours, caffeine and alcohol consumption), as well as an approval petition to collect the required data. The brainwave sessions are performed with 8 subjects, 5 males and 3 females, average age 25.125 years, standard deviation 1.54 years, range of 22-28 years. All subjects have normal vision, are right-handed and have no history of neurological disorders.

After the form is filled out, a short description of the procedure is given to the subject. They are only told that the objective of the agent is to reach the goal and the four movements that the agent can make. When this concludes, the subject is introduced to the wireless digital EEG device (g.Nautilus, g.Tec, Austria) that they have to wear during the brainwave session. It has eight electrodes (g.LADYbird, g.Tec, Austria) on the positions Fz, Cz, Pz, Oz, P3, P4, PO7 and PO8, identified according to the 10-20 International System, with a reference set to the right ear lobe and ground set as the AFz position. The electrodes contact points are adjusted applying conductive gel until the impedance values displayed by the program g.NeedAccess are within the desired range. This process takes between 10 and 15 minutes. After this step, the subject is instructed to close their eyes and the same program is used to check the live channel values so that there are no dead ones and the expected values are displayed for eye movements or muscle chewing.

Once the headset is correctly applied, the OpenVibe Acquisition Server program, from the OpenVibe platform [10], is launched and configured with a sampling rate of 250Hz and a 50Hz notch filter is applied to filter out power line noise as well as an additional bandpass filter between 0.5Hz and 60Hz. Data is handled and processed with the OpenVibe Designer platform using 8 channels for the brain data (one channel per electrode) and an additional channel for the stimuli. When everything is connected, the subject is seated in a comfortable chair in front of a computer screen. The brightness of the screen is set to the maximum setting to avoid any visual inconvenience in which the subject can not distinguish the components of the game that appear on the screen.

The Acquisition Server has the responsibility of receiving the signal data from the headset and any event information from the game and transfers it to the OpenVibe Designer application. When the subject is ready, the Game and the OpenVibe Designer programs are launched and configured to communicate with the previously mentioned acquisition server. A brainwave session consists of several experiences, each one being a game run, and it uses a pseudo-random function to determine the agent's movements. Once the experimental procedure is completed, the game state information and the signal data with the game event information of each run are saved. ACA For each run, the signals and stimulus information are then passed to the classification module that uses them to train the classifier. After this step is finished, the game movements and the trained classifier are used to update a Q-Table for each experience. Lastly, the calculated Q-Table is used to test if the agent has boosted its performance while playing the game. This module is explained in Section VI-A.

A. Cognitive Game experiment

The game parsimoniously consists of a $5x5$ grid of grey circular spots with a black background. A blue spot indicates current position of the agent whereas a green spot represents the goal, as shown in figure 4. The objective of the system is for the agent to reach the goal. The circular spot representing the goal remains static at the bottom-right position of the grid, while the one representing the position of the agent always starts at the upper-left position of the grid. When the agent reaches the goal, the position where the agent and the goal are located turns red, showing that the experience has ended. There are four possible movements that the agent can perform: it can go upwards, downwards, towards the left and towards the right, and those moves are bounded to avoid the agent from leaving the grid. The movement direction is selected randomly and is executed once every 2 seconds. When the experience ends, there is a pause of 5 seconds before the next experience starts. Each time an agent moves, it is considered as a stimulus to the observational human critic. The experience is designed for it to be evident whenever there is an error (i.e. the agent moves away from the objective) so the subject can notice it immediately after the stimulus is presented, possibly triggering the expected cognitive response, which can be imprinted as an ErrP component within the EEG stream.

III. Calibration

The first step in order to be able to find ErrP signals is to choose the most efficient algorithm, and the proper calibration. In order to do this different parameters are tested for a set of algorithms and for each individual subject. Initially a sub-selection of experiments is used to define a subset of parameters to test with, and then all the data is tested with this subset of parameters.

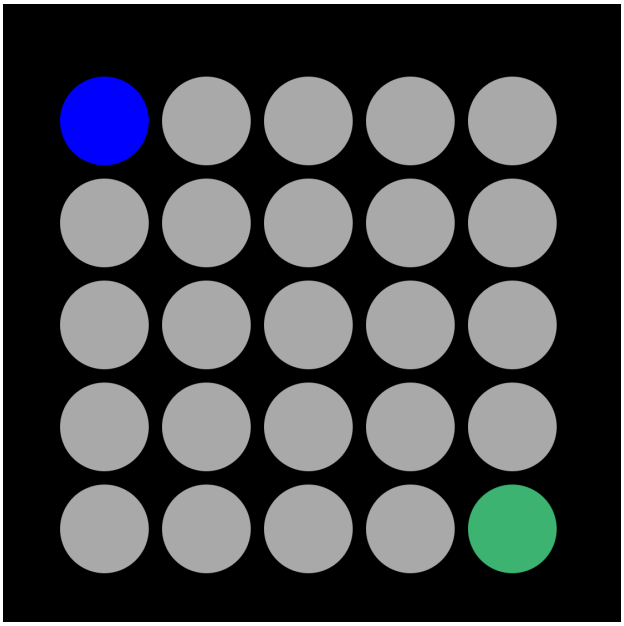
IV. Signal Processing and Segmentation

To aid in detecting the ErrP response, an offline processing pipeline and classifier is constructed to identify whether the action taken by the agent is an error or not. It is developed in Python using the MNE platform [11], which is a package designed specifically for processing EEG and MEG data, and the machine learning library Scikit-Learn.

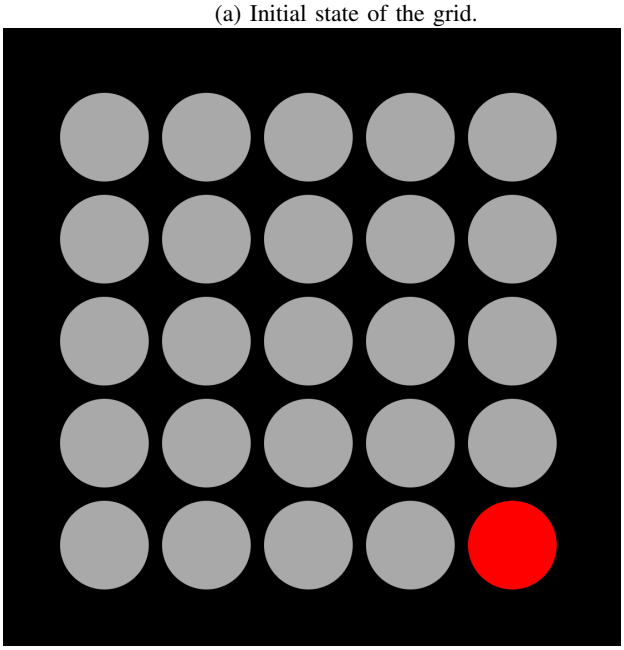
This step consists of offline processing the collected signals in order to train a classifier that can decide whether an error potential is triggered or not. Firstly, the output of a brainwave session is read and a band pass filter of 0.1-20.0Hz is applied to the signal. Samples that correspond to the start of an event are tagged using the data from the stimuli channel.

After the data is loaded and tagged, epochs are extracted from the raw data. Epochs consist of all the sample points that take place between the start of the event and 2 seconds later (time for each action to take place), resulting in 500 samples per channel, as the sample frequency is 250 Hz.

Samples that do not correspond to an epoch are not used. Also, epochs referring to the start or finish of the experience are excluded. This is done because the start and the end of the experience doesn't involve the agent taking an action,



(a) Initial state of the grid.



(b) Final state of the grid.

Fig. 2: Grid system representation used in cognitive experience.

thus giving signals that should not be considered in the classification process.

In this way, the raw data of an entire brainwave session is processed into an array of experiences where each element is an array of epochs tagged with a number specifying the prediction of the classifier, i.e. if the epoch corresponds to an action that made the agent move further from the goal (hit) or an action that made the agent move closer to the goal (no hit). The ErrP is expected to be found in hits. To get the data ready for classification, the stimuli channel is removed in order to classify the signals using only the EEG data. Each epoch is regularized using a MinMaxScaler. The eight channels are concatenated using the MNE Vectorizer function to transform the data into a single array sample. Lastly, this data is used by the classification module as information to train and test a classifier.

V. Classification

Vectorized epochs are used in the classification step. Four different classification algorithms are used. Logistic Regression, Multi-layer Perceptron with a hidden layer of 100000 neurons, Random Forest and finally a Support Vector Classifier (i.e. SVM) [**?**].

and finally the logistic regression classifier. Even tough the classification accuracy is low, the information is enough to have valid rewards that can be used to retrieve the information.

a classifier is trained for each subject and then epochs from their experiences are classified. These classifications are then used by a reinforcement learning algorithm to train the agent. This last algorithm is explained in detail in section VI-A.

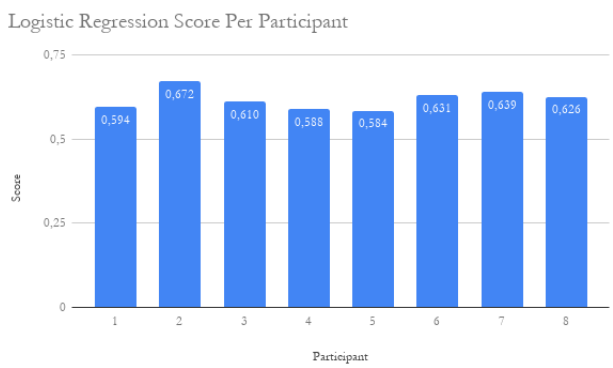


Fig. 3: Logistic Regression Algorithm Calibration

VI. Reinforcement Learning

The set of experiences of each subject is split into training and testing, so that the results of the classification can then be used to improve the performance of the agent. After training the classifier, the testing experiences are classified. Then, for each experience a reward file is generated. For this purpose, game states files, which are submitted by the game when an experience takes place, are used. These files contain an ordered list of the states of the game throughout an experience. With this file and the classification of the test data of an experience, a reward file is submitted. This file specifies a reward for each movement in the game, based on the classification of the event that corresponds to that movement. The reward can either be -1 when the event is classified as a hit or 0 when it is classified as a no hit. The accuracy of this rewards depends on the performance of the classifier. This reward file is used by the Q-Learning algorithm to improve its performance, as detailed in section VI-A.

A. Q-Learning

A Q-Learning algorithm is used to train the agent. It is developed in Python and uses the OpenAI Gym toolkit. Gym is

a toolkit for developing and comparing reinforcement learning algorithms. It makes no assumptions about the structure of an agent, and is compatible with any numerical computation library, such as TensorFlow or Theano.

1) Algorithm implementation: The Q-Table is initialized with zeros, unless a preexisting Q-Table is passed as a parameter. In order for the agent to learn from the feedback generated by the subject, the Q-Table is not used to determine the action to take at a given state. Instead, the policy which determines which action to take in a particular state is given by the steps taken from the brainwave session results. This allows to train the Q-Table based on the subject's feedback from the movements the agent took, which are chosen pseudo-randomly, while executing the brainwave session. The previously mentioned feedback is not explicit as it comes from the interpreted brain signal data, which is collected while the agent is executing the brainwave session and then each action is classified as an error or not. This implies that the reward is determined by the subject's brain activity. This is covered in further detail in **??**.

While using the previously mentioned step function, the Q-Table is updated in each iteration. This is done following the Equation 1.

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma * Max(s,a) - Q(s,a)] \quad (1)$$

After the algorithm finishes iterating through all the training episodes, the Q-Table is saved. Each experience is one $trainingEpisode$.

The reward function is calculated based on the current distance to the goal, if it has increased compared to the previous step then the rewards is negative, if not it returns zero. In order to test how the Q-Table is trained when the accuracy is not perfect the reward function doesn't always return the correct reward. It calculates a random number between 0 and 1, and if it is larger than the accuracy then it will return 0 even if it should return -1 as the reward.

The reward function is described in algorithm **??**.

```
reward ← 0
if currentDistanceToGoal ¿ previousDistanceToGoal then
    reward ← −1
else if random.uniform(0, 1) ¡ accuracy then
    return reward
else
    return 0
end
```

Algorithm 1: Reward Calculation for $ChaseEnv$

VII. Results

Enfatizar la idea de que la mejor métrica de la propia detección está dada por la propia capacidad del agente de minimizar su tarea, que hay un flujo de información que involucra la propia percepción de la persona.

A. Reinforcement Learning

Average steps to goal Figure **??** shows the average amount of steps it takes to reach the goal as the Q-Table is progressively trained using the reward information obtained for each subject. Each point corresponds to the average amount of steps it takes for the agent to reach the goal for a specific Q-Table in 200 iterations. The first point represents the amount of steps it takes to reach the goal for a Q-Table that hasn't been trained at all, where movements are decided randomly. The next point corresponds to the amount of steps it takes to reach the goal using a Q-Table trained with one experience, and so on.

The results show that as the Q-Table is progressively trained the average amount of steps decreases, meaning that the agent learns. However, the rate at which it learns varies per subject, certain subjects have more effective experiments than others, for example results for subject 1 (fig **??**) show faster learning than those of subject 8 (fig **??**).

In the case for subject 5 and 6, the reward information obtained from the brainwaves is not enough to train the agent effectively. Figures **??** and **??** show no apparent learning, as the amount of steps to reach the goal doesn't decrease when trained. Both subjects have less recorded data from the sessions in comparison to the rest of the subjects. In particular, subject 5 has less recorded experiences, so it is respective graphic shows the amount of steps for an empty Q-Table and for a Q-Table trained with only one experience. These results are consistent with fig. **??** and fig. **??**, which show that the signal classification for these subjects hasn't been particularly successful. The average steps result for these subject are similar to that of fig. **??** which is constructed from noise, so the generated Q-Table would be similar to one generated with noise.

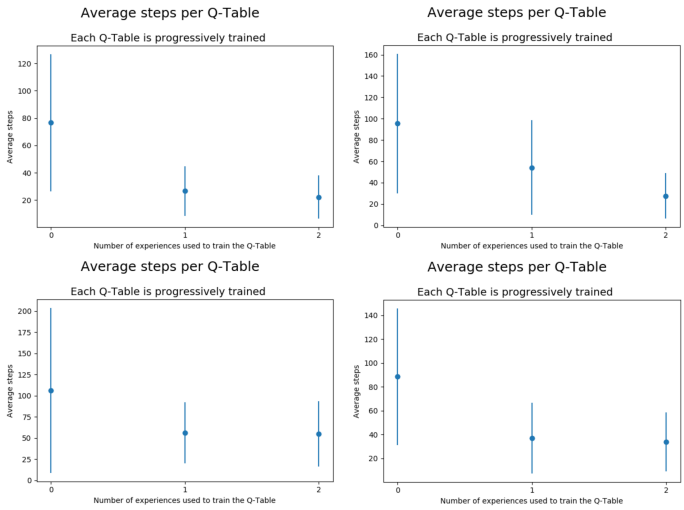
VIII. Conclusion

The present research work set out to validate if ErrP signals could be used to train an agent using reinforcement learning. The collected data show that ErrP signals can in fact be classified and used to train an agent effectively.

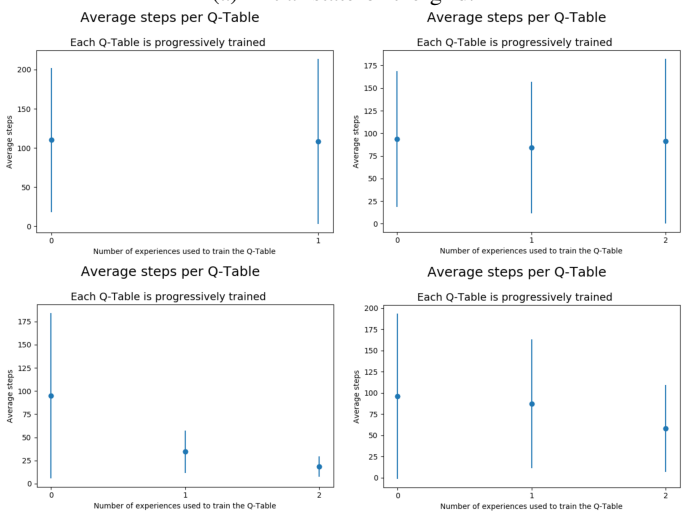
Enfatizar la idea de que la mejor métrica de la propia detección está dada por la propia capacidad del agente de minimizar su tarea, que hay un flujo de información que involucra la propia percepción de la persona.

When classifying, the better performing classifier is Logistic Regression. One important aspect of the classification results is the low percentage of false positives, meaning that it is not common that the agent learns that an action is wrong when in fact it is an action that takes it closer to the goal. On the other hand, the percentage of false negatives is generally higher, but this is not a serious issue since missing out on learning that an action is wrong does not lower the performance of the agent, but only means it will take more experiences to learn a correct path.

Once ErrP signals are identified they can be used to train a reinforcement learning algorithm. However, this can only be achieved by generating a specific classifier for each subject and using it for giving rewards for the corresponding subjects. Results show that training a classifier with data of one subject,



(a) Initial state of the grid.



(b) Final state of the grid.

Fig. 4: Grid system representation used in cognitive experience.

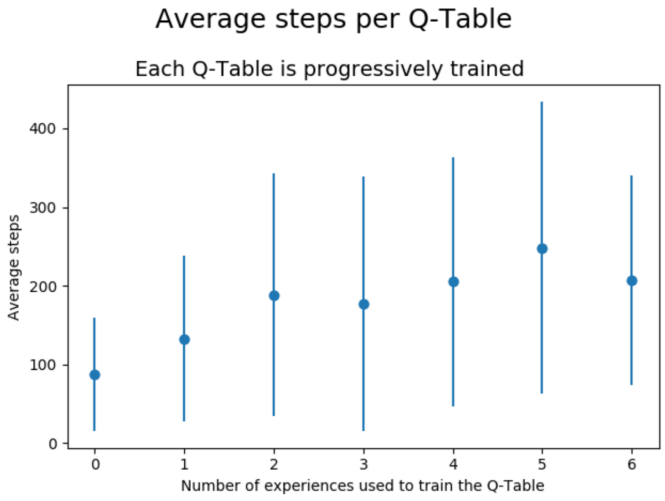


Fig. 5: Average steps using Q-Table trained with noise.

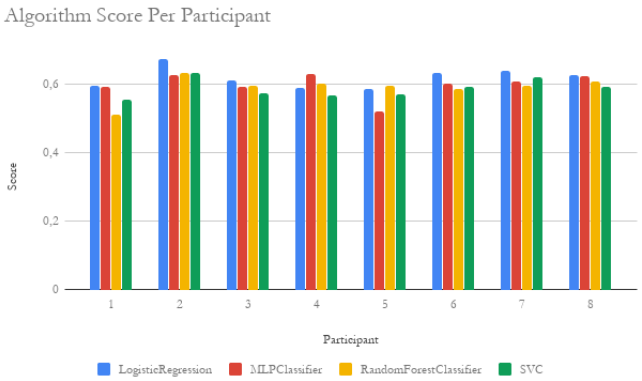


Fig. 6: Score Obtained With The Optimal Calibration Per Algorithm Per Subject

but using it to classify the events of experiences of another subject does not lead to an improvement on the performance of the agent, concluding that the experiment is not suited for transfer learning. Despite that, the rewards generated from different subjects can be used to train the same Q-Table to improve its performance.

Brainwave sessions have a low amount of experiences in order to reduce fatigue from the subjects. However data suggests that longer sessions are required in order to reach better classification scores, since more data is available in order to train the classifier. It can be seen that subjects with the largest amounts of data have the best classification. This can also be achieved designing a bigger game system that generates more samples with every session.

Finally, this research verifies that brain signals can be used as an interface between human and computer enabling the control of the system without explicit input from the user.

Acknowledgment

The authors would like to thank...

References

[1] N. P. B. Thorsten O. Zander, Laurens R. Krol and K. Gramann, *Neuroadaptive technology enables implicit cursor control based on medial prefrontal cortex activity*, 2016.

[2] C. B. Holroyd, O. E. Krigolson, R. Baker, S. Lee, and J. Gibson, "When is an error not a prediction error? An electrophysiological investigation," *Cognitive, Affective and Behavioral Neuroscience*, vol. 9, no. 1, pp. 59–70, mar 2009. [Online]. Available: http://www.springerlink.com/index/10.3758/CABN.9.1.59

[3] P. Ferrez, "Error-related eeg potentials in brain-computer interfaces," 2007.

[4] R. Chavarriaga, A. Sobolewski, and J. d. R. Millán, "Errare machinale est: The use of error-related potentials in brain-machine interfaces," *Frontiers in Neuroscience*, vol. 8, no. 8 JUL, p. 208, jul 2014. [Online]. Available: http://journal.frontiersin.org/article/10.3389/fnins.2014.00208/abstract

[5] K. S. I. A. A. H. A. G. T. H. L. B. M. L. A. B. Y. C. T. L. F. H. L. S. G. v. d. D. T. G. D. H. David Silver, Julian Schrittwieser, "Mastering the game of go without human knowledge," 2017.

[6] J. M. I. Iturrate, L. Montesano, "Robot reinforcement learning using eeg-based reward signals," 2010.

[7] J. Omedes, I. Iturrate, L. Montesano, and J. Minguez, "Using frequency-domain features for the generalization of EEG error-related potentials among different tasks," in *Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBS*. IEEE, jul 2013, pp. 5263–5266. [Online]. Available: http://ieeexplore.ieee.org/document/6610736/

[8] L. Schiatti, J. Tessadori, N. Deshpande, G. Barresi, L. C. King, and L. S. Mattos, "Human in the Loop of Robot Learning: EEG-Based Reward Signal for Target Identification and Reaching Task," in *Proceedings - IEEE International Conference on Robotics and Automation*. IEEE, may 2018, pp. 4473–4480. [Online]. Available: https://ieeexplore.ieee.org/document/8460551/

[9] D. Plass-Oude Bos, B. Reuderink, B. van de Laar, H. Gürkök, C. Mühl, M. Poel, A. Nijholt, and D. Heylen, "Brain-Computer Interfacing and Games." Springer, London, 2010, pp. 149–178. [Online]. Available: http://link.springer.com/10.1007/978-1-84996-272-8{_}10

[10] G. G. M. C. E. M. V. D. O. B. Yann Renard, Fabien Lotte and A. Lecuyer, "Openvibe: An open-source software platform to design, test and use brain-computer interfaces in real and virtual environments," 2010.

[11] E. L. D. E. D. S. C. B. R. G. M. J. T. B. L. P. M. H. A. Gramfort, M. Luessi, *MEG and EEG data analysis with MNE-Python, Frontiers in Neuroscience, Volume 7*, 2013. [Online]. Available: https://mne-tools.github.io/0.13/index.html

PLACE PHOTO HERE

Michael Shell Biography text here.

John Doe Biography text here.

Jane Doe Biography text here.