Introduction

As computer development branches into three dimensional interfacing, there is significant interest to develop alternative methods of input. Through significant reclassification of computer interaction, it may be possible to create methods of input which were not previously possible. One such interaction method is that of Brain Computer Interfaces (BCIs). These devices allow for electrodynamic action potentials in the brain to be translated into computer signals for input classification.

The concept of creating a classifier for actions works on the principle that each action produces a unique pattern in the signals of the brain. These signals are repeatable, meaning that each action produces the same unique signal every time the action is performed. As a result, if multiple actions are performed, each action will produce a unique signal to be used in the classification process.

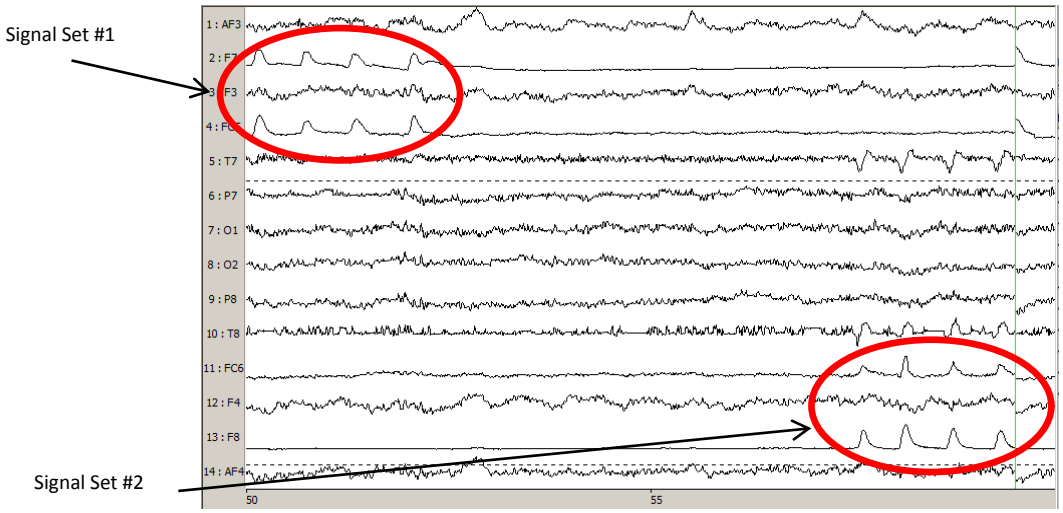
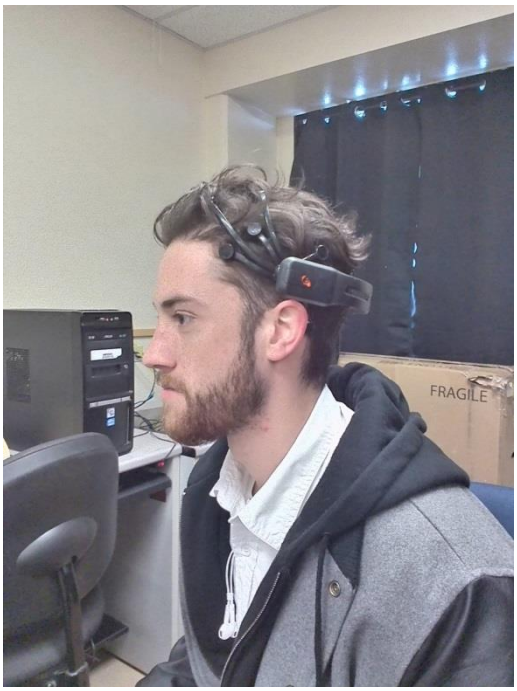


Figure 1: BCI displaying two separate actions performed

Experimental Details

The BCI device used in all testing was the Emotiv EPOC, a sixteen sensor device. Two of the sensors acted as "guides", indicating whether the headset was properly worn. This device was placed on the user's head, allowing for 14 bands of signals to be passed to the computer through Bluetooth. Each signal is then passed through the OpenViBE software, which broke the signals into define "chunks". These chunks were large matrices of data, each containing fourteen rows indicative of the fourteen

bands read from the user. One hundred twenty eight of these chunks were generated per second of data passed to OpenViBE.



Picture 1: User demonstrating alignment of Emotiv EPOC

Each chunk of data was passed through OpenViBE to a corresponding MATLAB script, which depended on whether the user was classifying input or using the headset for interfacing. In the classification script, the user was asked to press a key on the keyboard while performing an action to produce a unique signal. As the key was pressed, the script checked if the chunk contained components that were significantly different from the background signal. If significant difference was detected, the features were passed into a dictionary like object which mapped the features to the keypress. The user was then asked to repeat this process for every key mapped (usually 2-3 keys).

When the gathering of data to be classified was halted, the MATLAB script would automatically generate a neural network based upon the number of keys used, and an additionally output for a non-input signal. This neural network is then saved to the system for later use in the interaction script.

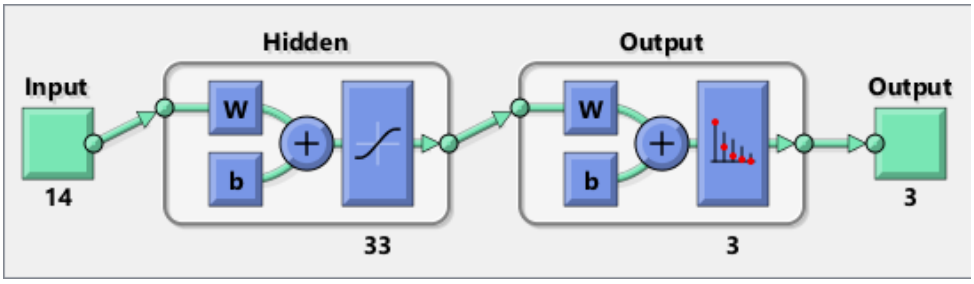


Figure 2: Representation of a MATLAB neural network with a 2-key output. Note that the input value of 14 indicated the number of bands of data gathered.

The interaction script behaved similarly to the input script in the acquisition of data. The Emotiv EPOC headset passed data into a separate OpenViBE program, which then passed each chunk into a MATLAB script. The interaction script loaded the previously saved neural network. Each chunk passed into the script was quickly searched to obtain possible features. If no features were found, the chunk was ignored. The isolated features were then passed through the neural network, which gave a probability of which output the features corresponded to (either one of the possible key presses or a non-input). If the signal was determined to be a key press (that is, if the output of the key press option from the neural network was above a certain probability), the associated key press was input into the system.

Results

During the process of programming the MATLAB script, various numbers of nodes were used for the neural net. While significant change between nodes was not found at high numbers of nodes, the amount used for testing was settled around 33 nodes. The system was also highly sensitive to how many data points for each point were recorded. To ensure efficacy of the MATLAB scripts, each key press was repeated at least 10 times.

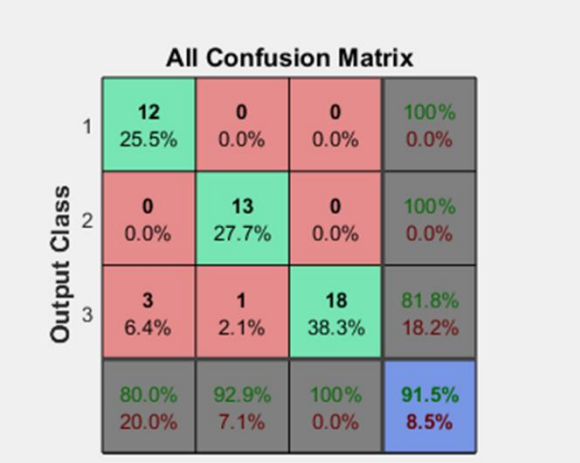


Figure 3: Confusion matrix for a 2-key press system.

Figure 3 demonstrates the efficacy of the program in a 2-key press system. Output value 1 and 2 are the associated key presses, where output 3 represents a non-input. Non-input data was gathered periodically by the script in order to properly generate the value. The overall correctness of the system, shown in the bottom-right blue box, shows that the system is relatively correct. This value varied depending on the number of data values entered and the number of nodes used in the neural network.

Figure 4 displays the Receiver Operating Characteristic (ROC) graph for the same neural network as figure 2. The graph shows that the system often performed better than a random-guess output, with the variation being in the actual key-press functions.

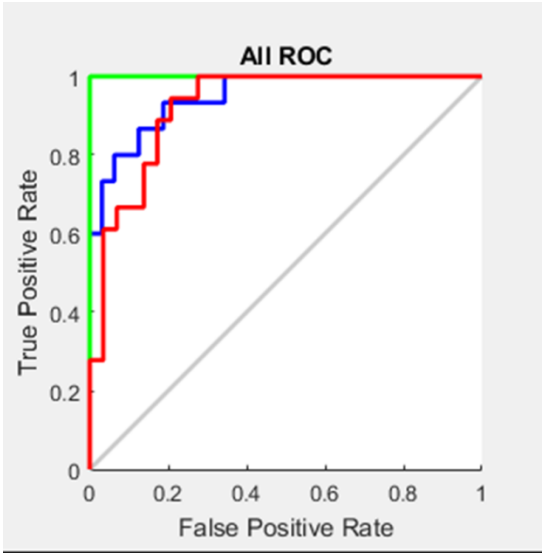


Figure 4: ROC graph for a 2-key press system.

Figure 5 displays the minimization of cross-entropy over epochs of training. Each training epoch is plotted with relation to the neural network output "correctness". The minimum value of the green line (which indicated the validation set of data) is ultimately the system state that is saved by the program, as epochs after this point do not show significant improvement.

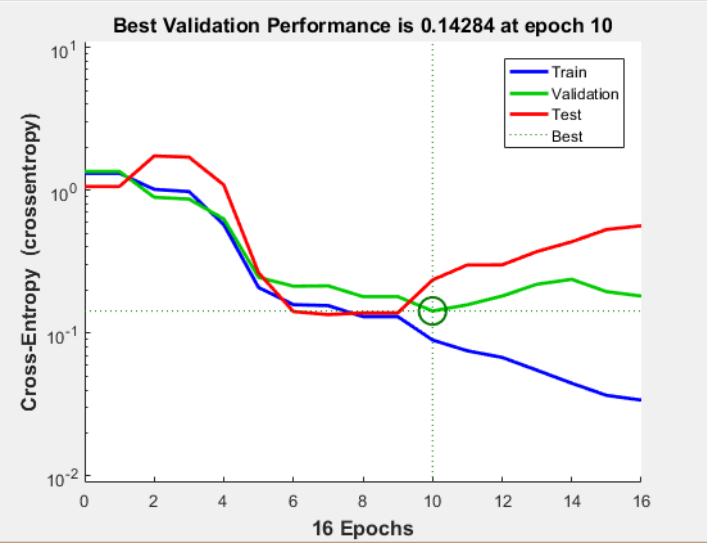


Figure 5: Validation graph for a 2-key press system.

Discussion

The system performed with decent accuracy in two and three key press systems. In usage of the resulting neural networks, high accuracy was observed in correlated movements and output. Error exists in the system through a combination of the headset, user input, and neural network classification.

The headset was error prone due to the Bluetooth passing of data. Bluetooth struggled to accurately track the headset position unless there was an uninterrupted path between the headset and the Bluetooth dongle. The headset also rarely had all 14 trackers accurately tracking, if tacking at all. While this did cause some error, the headset is designed to work with a lesser number of trackers providing data. It was often the case that unique data could be obtained for each movement despite this, but accuracy may be improved if all trackers worked.

The system was also prone to error if the user did not correctly synchronize actions and the training key press. If the classification script did not have any method to determine if the data that was passed matched previous data associated with that key press, which occasionally led to data being associated with a key press that was incorrect. This led to occasional misclassification of data, either through non-inputs being classified as key presses or key press actions not registering. This could likely be reduced if the script was updated to check if the data passed on a key press match similar data that the key press had already been associated with. It is important to note, however, that any action involving parsing of the data was often computationally expensive. In order to reduce latency in user action, this was determined to be an acceptable error.

Finally, the neural network was often sensitive to the number of data points passed for each key press. If a low number of key presses were used, it was found that the system was highly inaccurate, and often had misclassification of key press actions. The system performed more accurately with a large number of data points for each key press. This was, however, tedious for the user to train, as they would need to repeat an action multiple times for each data set. The larger the data set, the more times the action would need to be repeated. This may be tiring or difficult for the user to do. As a result, the system was usually trained on around 10-20 data points for each key press.

It was also noted in the course of this experiment that users had occasional difficult in repeating an action exactly as the neural network had been trained. As a result, performed actions that were thought to produce a key press occasionally did not produce the output expected. This could be reduced if the user was better trained to perform the repetitious action, or if the script was better and recognizing similar actions and producing the same output for each.

Conclusion

Through a series of programs, a successful classifier for mapping actions to key presses was created. This system collected data from the user in a series of actions mapped to key presses, which

was then passed to a neural network. After training the neural network, it was saved to the system for usage of input classification. The user was then able to use an action to produce an associated key press.

This system was fairly accurate, and often was able to reproduce key presses associated to each action in training. While the system did suffer from errors in the equipment, user input, and neural network training, overall these errors did not prevent the program from achieving accurate output. The user classification of data was the largest source of error, as the system was susceptible to inaccuracy if the data set for training the neural network was not accurate. The systematic error of the inaccuracy of the headset was determined to be within acceptable ranges, though it was noted that the system could be improved if more trackers were working. The neural network suffered occasional error, but the error inherent in the network was often less than error produced through inaccurate user classification of data.

This system offers several possibilities for future research and usage. While the headset used was not accurate enough to map fine motor actions, it is possible that other BCI devices can track these motions. Should this be the case, BCI devices may be a gateway to various gesture recognition applications. Additionally, actions are known to produce a similar signal if the user merely thinks about performing the action. If it were possible to successfully classify these signals, BCIs may provide interface possibilities for low-mobility and disabled users.