Sensors & Transducers, Vol. 257, Issue 3, May 2022, pp. 30-37

www.sensorsportal.com

Sensors & Transducers

Published by IFSA Publishing, S. L., 2022
http://www.sensorsportal.com

Brain-Robot Interface Based on Artificial Intelligence

Ayodeji Onipe, Amiel Dominic Gozon, Eric Yang, Ishwar Singh and * Zhen Gao

W Booth School of Engineering Practice and Technology, McMaster University
Main St W, L8S 4L8, Hamilton, ON, Canada
E-mail: gaozhen@mcmaster.ca

Received: 18 April 2022 /Accepted: 23 May 2022 /Published: 31 May 2022

Abstract: The purpose of this research is to be able to fully control a robot's movement from a user's thought and movement. The OpenBCI (Open□Source Brain Computing Interface) is applied that allow for electronic communication from different parts of the human body. In the case of this implementation, the source will be the brain. The acquisition comes in the form of an electroencephalogram or EEG which will output an electric signal from specific nodes touching the brain. These signals come in numeric form which will be used to train a neural network which will be able to distinguish between a user's thoughts and hand movements.

Keywords: Brain-robot interface, EEG, Artificial intelligence.

1. Introduction

Brain-Computer Interface has been extensively used in various aspects of industry, healthcare, and social life [1-7]. In the area of Brain-Robot Interface, scholars also conducted explorations in theory and practices [8-12]. In the area of brain-interface design, scholars explored the applications in telepresence, care of spinal cord injury, and remote robot arm [13-15]. For this research, the goal is to be able to control a light-weight legged robot with thought by using electroencephalogram (EEG) signal pulses). Specific directional thoughts and arm movements will be the main measurement and deliverable of the data extraction.

There will be a variety of software and hardware used to compile and interpret the data. In terms of hardware, the OpenBCI Ultracortex Mark IV Helmet fits onto a user's head and comes attached with 8 nodes to extract the EEG signals from the brain. On top of this, the OpenBCI Cyton Board is attached to the helmet and all the nodes are routed to the pins on the board. The artificial intelligence algorithms that will applied to interpret the data from the Cyton board. A graphic user interface will be developed that can control the robot and will be equipped with the algorithm to make predictions. Ideally, the user will think of a direction and the robot will direct itself to it by turning its motors forward and backward.

2. System Design

2.1. Original Design

The hands-free human-to-machine interface using EEG signals and a small-scale mobile robot are developed in this research. The objective was to prove and test the performance and reliability of EEG signals as a method of hands-free machine operation.

The project is a bold step into uncharted territory. It is data intensive and relies heavily on approximations of brain activity gathered through an 8-channel EEG signal headset. The research entails a real-time, practical application of data processing,

translation, and machine actuation. Video information will be sent to the user so that they may effectively steer the robot without having the robot in their line of sight. Using the video input from a camera some ambiguity around control will be clarified as the user will only have to worry about left, right, forward, and backward from the vehicle's front perspective.

The headset has 8 nodes which are the components used to collect EEG data for our models. The Cyton board is also capable of collecting data acceleration data which refers to the position of the headset in regard to the x, y and z planes. The GUI is what we use to ensure that all the nodes of the headset are working correctly before we collect our data samples.

The data collection has some inconsistencies as the connection between the dongle and Cyton board can have some issues. This leads to a differing number of rows for the total amount of data collected within the same time range. A 5 second data sample can range between anywhere from 900 to 1200 rows of EEG signals. To call the program, we must pass through 2 parameters which are the board ID and serial port. The board id indicates which OpenBCI board we are using and, in our case, the Cyton board with 8 channels is ID 0 and the serial port just depends on the computer.

We collected a total of 104 data samples for each thought category with a total of 312 samples collected in total. The data collection is stored in NumPy arrays and is based on time connections with the board which leads to some inconsistencies which is why we collected data for 5 seconds but sliced the data to ensure all the data samples are the same size with the 8 channels of EEG signals and 700 rows of signal data. One thing to keep in mind is that the samples for the left thought were collected all in one session and the right and other thoughts were collected all in another session. This could cause the data to be less accurate as each session the headset tends to perform differently.

All the collected data is appended into a single NumPy array and separated by a label that indicates which action it is associated with. The left thought is labeled with a [1,0,0], none is labeled with [0,1,0] and right is labeled with [0,0,1]. This NumPy is shuffled to help the model learn better later and the features - NumPy array of sample and labels - the action thoughts are stored in their own separate variables X and y. The data is reshaped just in case, however they were already reshaped before during the collection process and saved so it can be passed into the convolutional neural networks model later.

Our initial design consisted of us collecting data samples for 3 thought intentions and trying to train a convolutional neural network to test if it was possible to have a model predict our intentions.

2.2. Evolved Design

Our original idea was to use a RC car to demonstrate our machine learning model's predicted outcome but have now moved to using a more complicated robot. We are currently on our improved design that works well. The bot uses 8 servos in total, 4 for legs and 4 for hips. Using a HC-06 Bluetooth module, it could receive data from python and takes those commands to perform a corresponding action matching to one of the five thought actions we are trying to predict. Fig. 1 shows the prototype of the light-weight legged robot.

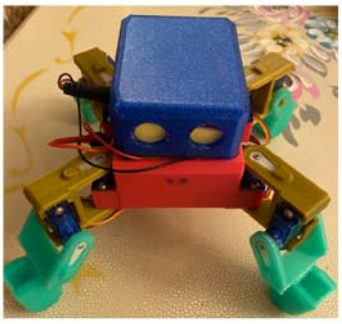

Fig. 1. Light-weight legged robot.

Formerly, only three directional intentions were sampled for about 5 seconds. Then the instances were trimmed and packaged so that the data would be more uniformed, and they were approximately about 3 seconds worth per instance; each instance represents a single data collection run for a particular directional intention stored in a 2-dimensional NumPy array (700 rows by 8 columns). The adapted data collection method sampled directional intentions for 5 commands: Forward, Right, Backward, Left, and Other – the do-nothing command.

During data collection, the subject remained completely still. This was to minimize noise and artifacts in the samples as much as possible. No arbitrary tasks were performed, and the subject was seated through each sampling period. During the sampling runs, the subject chanted the directional intention in their minds (Left, Left, … etc.), while picturing a left directional icon, through the duration of the sampling period.

In retrospect, this was not the most verifiable method of data collection but at the time minimizing noise and staying true to the objective – not relying on any other signal except EEG – was paramount. Any hand movement, or eye movement that was not filtered out would have contaminated our samples and disrupted our goals.

3. Implementation of AI Approaches

From our initial design to test our concepts, we had collected data for 3 thoughts (left, right and other) to train a convolutional neural network to predict our thoughts. We have now changed from 3 thoughts to 5 (backwards, forward, left, right and other), each with 150 samples of collected data. In addition to changing

the number of thought samples, we have also changed our approach to how the samples were collected. This time around, the samples were collected over 15 different sessions and in each session, 10 samples of each thought were taken. This should give us a better dataset to train and validate our models. We have also branched out to explore more models in addition to the convolutional neural network such as random forest, support vector machine, extreme gradient boost (XGBoost), backpropagation, Naive Bayes and k-nearest neighbor. The training and testing datasets we have setup are the same for all the models except for the convolutional neural network. Convolutional neural networks do feature learning first on the data before it is flattened into a 1D array while other models will only need the data in its flattened form to train and test. For these models, we are still tuning parameters and adjusting our data to further optimize our model accuracies. We will also try to branch out into exploring unsupervised learning methods such as k-means clustering.

Fig. 2 shows the normalized confusion matrix of the convolutional neural network. The diagonal shows evidence of fair-good generalization on 'unseen' data. The worst performing class is Left, hinted at from the low ratio in confirmation with the prediction. A significant amount of the time left is misclassified as right. This is more evident in the absolute confusion matrix where those mistakes are emphasized.

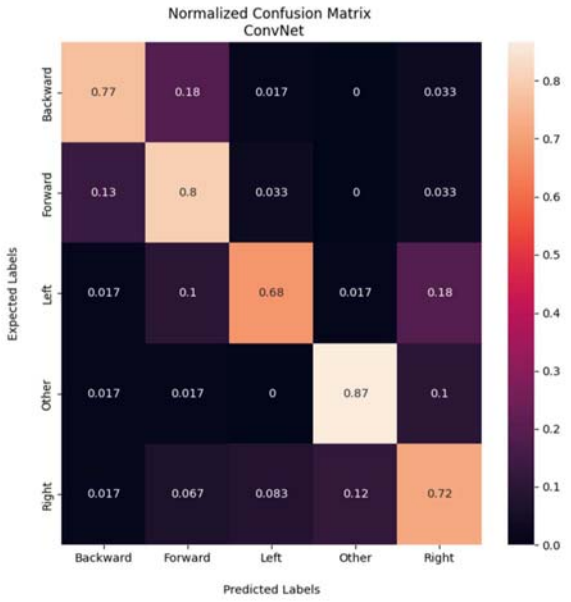

Fig. 2. Conv-net normalized confusion matrix.

Fig. 3 is the absolute confusion matrix which emphasizes the misclassifications by blanking out the diagonal (representing recall). This allows the scale to adjust and provide more emphasis to the discrepant misclassifications. Notice that both Left is misclassified as Right, and Backward is misclassified as Forward almost a fifth of the time.

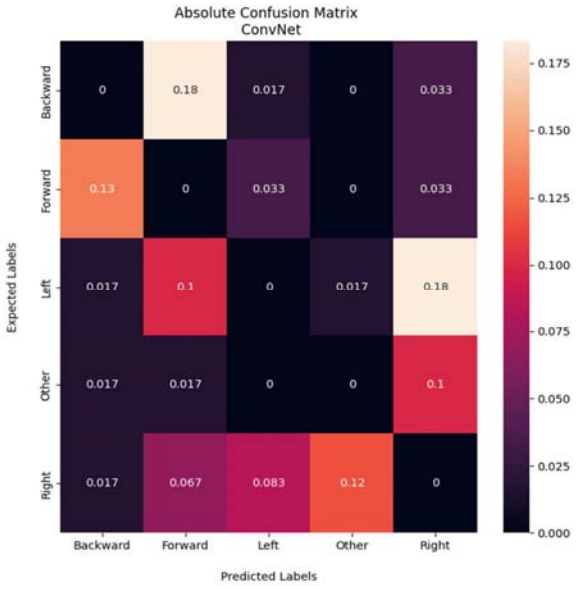

Fig. 3. Conv-net absolute confusion matrix.

The normalized confusion matrix for the support vector machine algorithm is attached below (Fig. 4). It performs slightly better in comparison to convolutional neural networks. The diagonal is clear, this is evidence of confident recall.

Similar analysis of the absolute matrix below (Fig. 5), highlights Forward - Backward and Left – Right misclassifications. These are the mistakes this particular SVM algorithm is likely to make in a live run.

XGBoost was evolved from decision tree. The XGBoost confusion matrix shows fair recall (see Fig. 6). This is represented by the fuzzy valued diagonal bordering on the percentages 78-90 %.

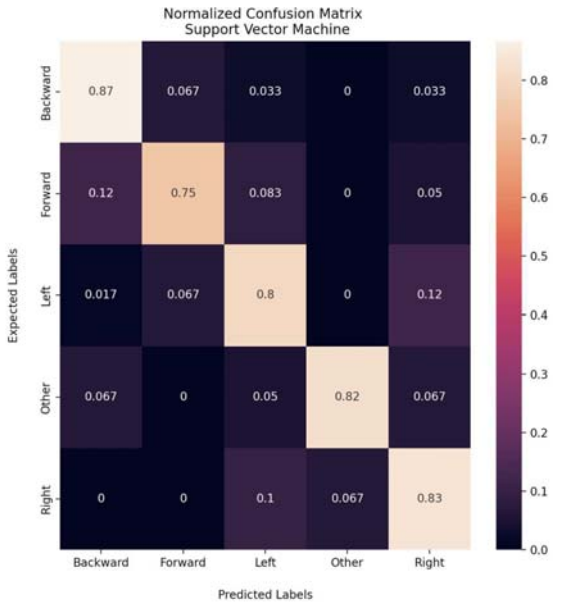

Fig. 4. SVM classifier: normalized confusion matrix.

The XGBoost algorithm is likely to make Forward - Right misclassification mistakes during live runs. There are also other common misclassifications this algorithm would be prone to, as shown in Fig. 7.

The Normalized confusion matrix for the K-Nearest Neighbour (KNN) algorithm is attached below (as shown in Fig. 8). Notice that the diagonals are sharp. This model performs very similarly to the SVM.

The absolute confusion matrix blanks out the diagonal to emphasize the models' common mistakes. This model's most common is misclassifying Backward as Forward, as shown in Fig. 9.

The random forest algorithm, our best performing algorithm thus far, boasts the clearest diagonal. It is close to zero values for misclassifications, as shown in Fig. 10.

In the absolute confusion matrix, by blocking out the diagonal, and resetting the color scale the common mistakes are emphasized. This model is prone to misclassifying Forward as Backward, and Right as Other. Those are the most common mistakes to expect using the model live (see Fig. 11). From the classification report as shown in Fig. 12, it is found that the overall performance of Random Forest is great.

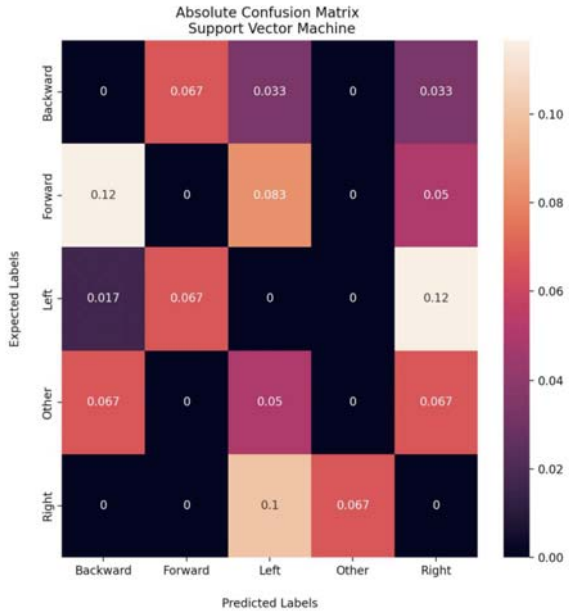

Fig. 5. SVM classifier: absolute confusion matrix.

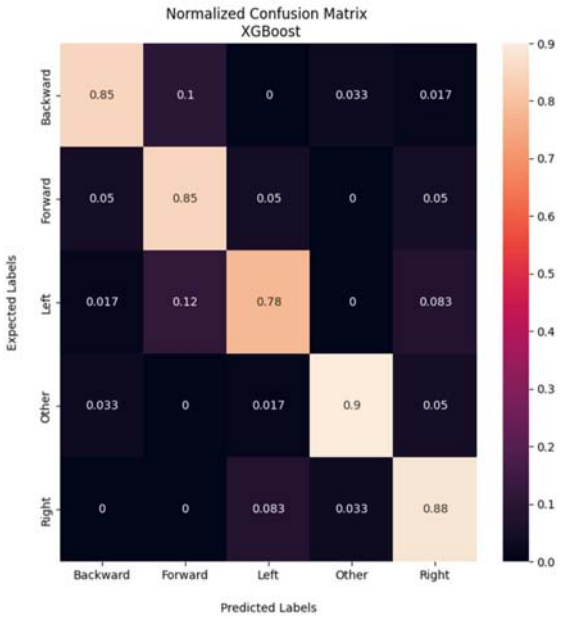

Fig. 6. XGBoost: normalized confusion matrix.

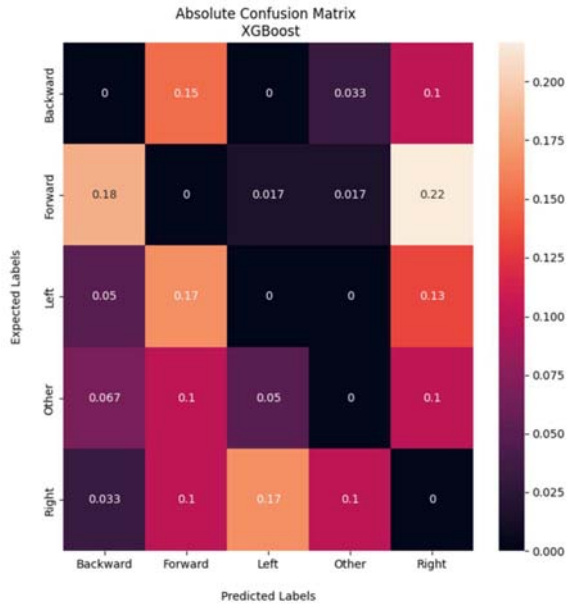

Fig. 7. XGBoost: Absolute Confusion Matrix.

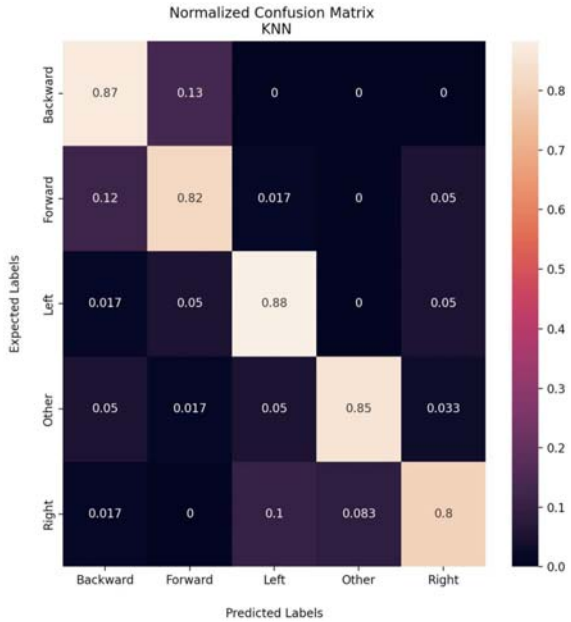

Fig. 8. KNN: normalized confusion matrix.

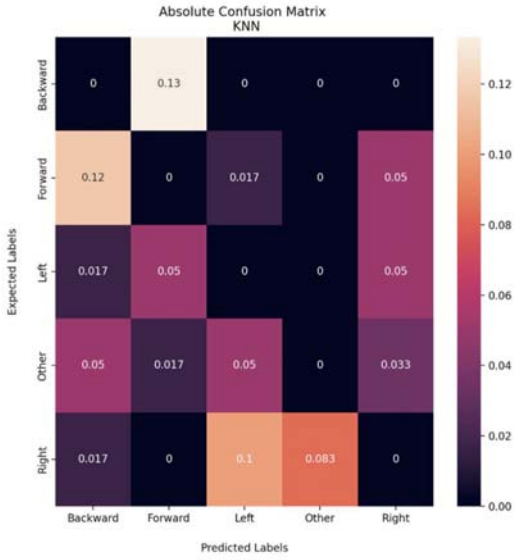

Fig. 9. KNN: absolute normalized confusion matrix.

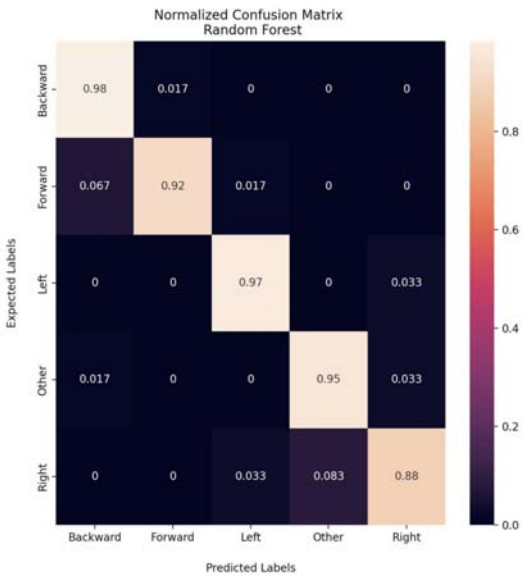

Fig. 10. Random Forest: normalized confusion matrix.

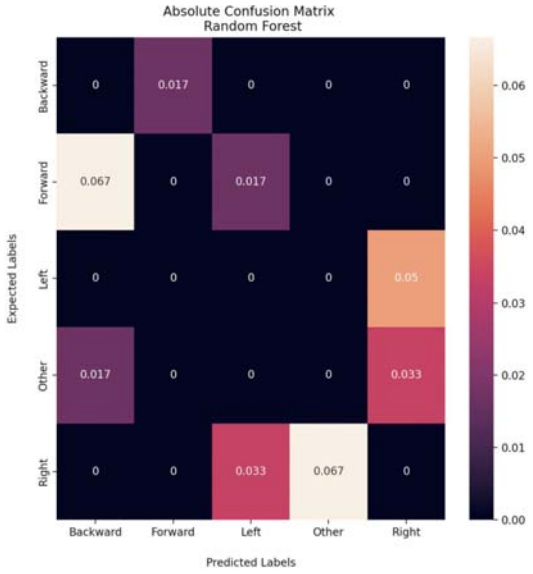

Fig. 11. Random Forest: absolute normalized confusion matrix.

	precision	recall	f1-score	support
0	0.92	0.98	0.95	60
1	0.98	0.92	0.95	60
2	0.95	0.97	0.96	60
3	0.92	0.95	0.93	60
4	0.93	0.88	0.91	60
accuracy			0.94	300
macro avg	0.94	0.94	0.94	300
weighted avg	0.94	0.94	0.94	300

Fig. 12. Random Forest: classification report.

Fig. 13 below shows the confusion matrix of the naive Bayes algorithm. Notice, unlike the other models tested it does not show a clear diagonal. This means that the recall is poor. It also makes a mess of classifying the test set appropriately; significant rations of the set are badly mistaken for each other. More staggering is evidence of anti-Right behaviour, the model seems to avoid predicting any of the instances as Right. Right is the last label, the team suspects this has something to do with this oddity.

Fig. 14 below shows the absolute confusion matrix of the Gaussian Naive Bayes algorithm. This model makes by far the most mistakes in misclassification. Notice that the model misclassified Backward as Forward more than half of the time. Right as Other just less than half of the time.

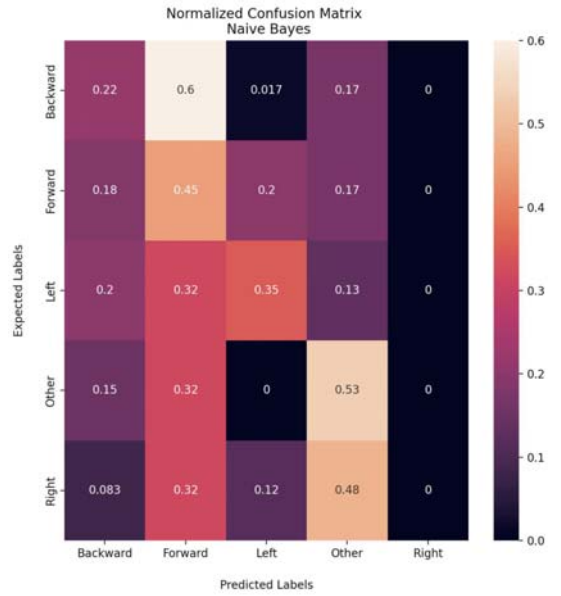

Fig. 13. Naive Bayes: normalized confusion matrix.

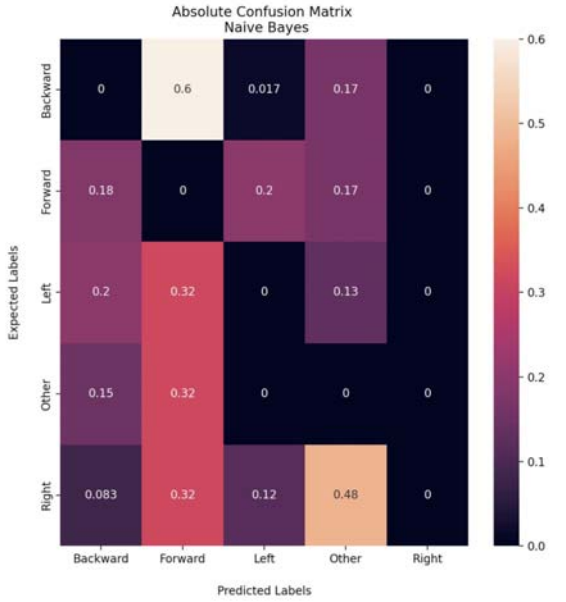

Fig. 14. Naive Bayes: Absolute Normalized Confusion Matrix.

4. Discussion of AI Implementations

Fig. 15 shows the F1 scores of all the models evaluated thus far. The best performing is the Random Forest algorithm, in theory it performs very well on 'unseen' data and has a very impressive confusion matrix. The support vector machine and k-nearest neighbor algorithms perform similar well, both tied at approximately 84 % in theory. The convolutional neural network and XGBoost are fairly good.

However, notice how the Naive Bayes significantly underperformed relative to the pack. This is not to say that it is a horrible algorithm or model, rather it goes to show it is simply not suitable for our application. The model just does not lend itself to situations where features intercorrelate significantly.

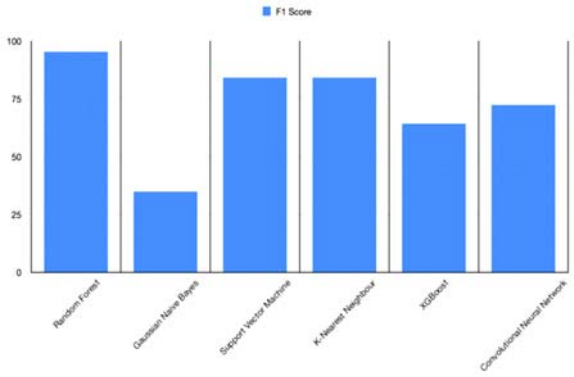

Fig. 15. F1 score comparison plot.

As these models mentioned above provided results in terms of their F1 scores, they did not perform well when it came to using them to predict actions in a live setting. This could be attributed to many factors such as the models being overfit, the time gap between taking values and testing, and absence of data filtering. From this, we looked at more research conducted with processing EEG data and found examples of using a low-pass filter such as a rolling average filter to smooth out the data as well as subsampling to help process the data first. So, we implemented a rolling average and subsampling, both individually and in combination with each other and then used that new data in our models.

With the Data being filtered, we used the new datasets (including datasets where only rolling average was used, only subsample, and a combination of both) to train our models and did not get any good results from it. Since we were looking more for a proof of concept that applying this would help, we conducted the testing on the KNN model as the validation check was faster than other models. We tried various number of values for the rolling average on the KNN model for faster results and were able to see that if the rolling average was greater than 5, the model tended to perform poorly as shown in Fig. 16.

However, even with the rolling average of 5 performing well on its own and with subsampling, when it came to testing with live data, the model was overfitting and unable to predict the thoughts, as shown in Fig. 17. This again could be affected by the time gap between the data collection period and the testing period, or it just was not enough to significantly help clean the data.

Training the Convolutional Neural Network with our new dataset gave us mediocre but better than random validation values. The validation accuracy peaked at about 40% and our validation AUC score peaked at about 67%. If we look at these values with a fresh point of view, these values show good potential for the model to predict the different thought actions at a decent rate. From Fig. 18, we can see the training

and test sets of data start to separate roughly around 10 epochs which is generally when you'd want to stop training the model, but we based our model on the validation AUC score and stopped training once it reached a peak score and the next five scores all are less than the peak. The confusion matrix shows us that the model performed relatively well when it came to predicting backward, forward, and right but struggled with the left action thought. With more tuning, it could be improved but this shows a lot of potential in performing well.

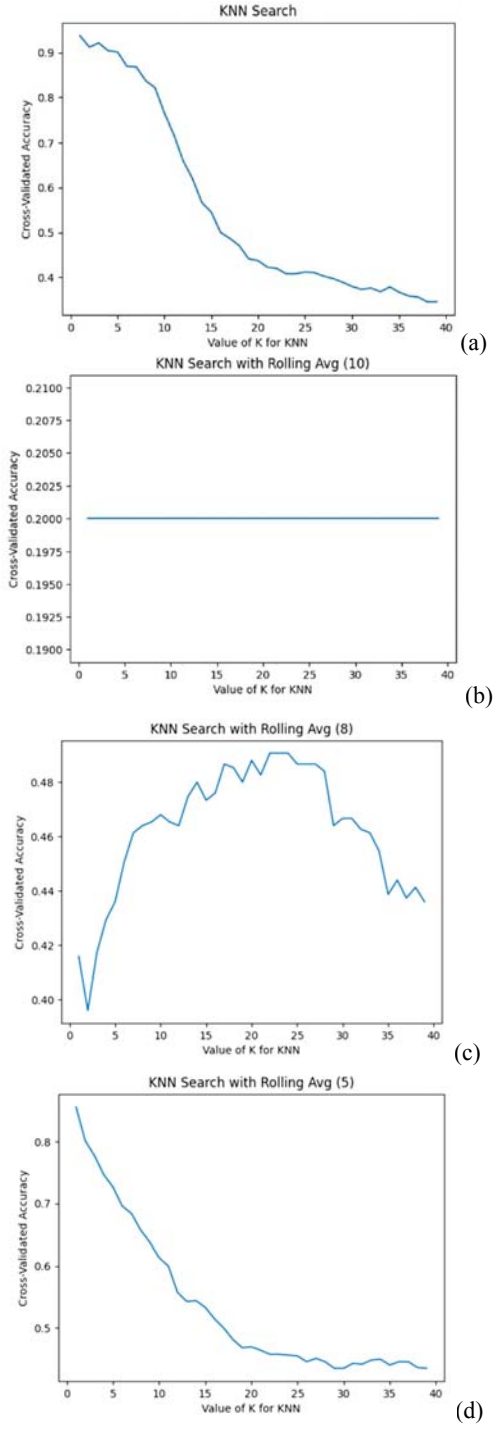

Fig. 16. KNN Search with various parameters for preprocessing.

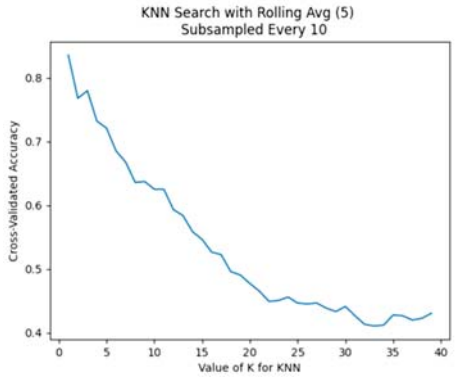

Fig. 17. KNN model search with rolling average of 5 and subsampling of 10 on the data.

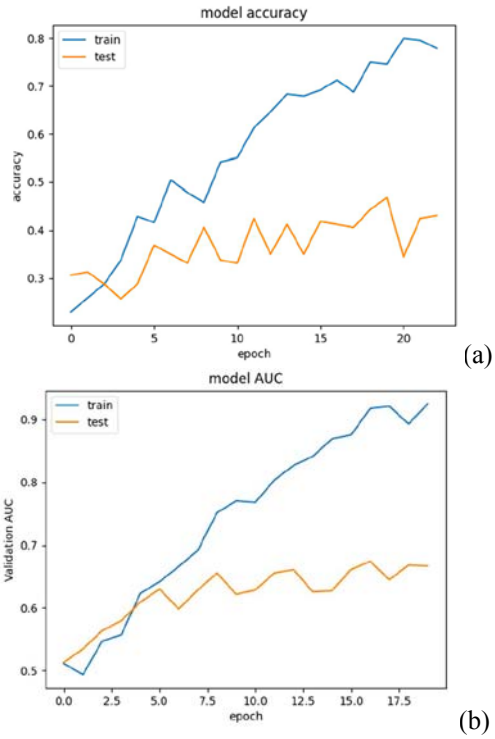

Fig. 18. On the left we have the CNN accuracy vs epoch and on the right, is the validation AUC score vs epoch.

5. Conclusions

The objective of this research was to develop a brain computer interface that would translate EEG signals into directional commands to action a mobile robot in real time. Although the model can distinguish directional commands based on incoming EEG signals it is far from perfect and quite inconsistent, for that reason a GUI which supported both handsfree and remote-controlled modes was developed in parallel. For practical applications of brain computer interfacing, although the project is at its end, there is still room for improvement. It is important to keep in mind that it may be better to test and train the models within the same headset streaming session rather than having to restart the headset streaming which can

interfere with some of the early sections of the data. Either that or, to train and test a model within a shorter time span as gaps between these two events have proved to be a major factor in the overall accuracy of the models.

References

[1]. American Neethu Robinson, Ravikiran Mane, Tushar Chouhan, Cuntai Guan, Emerging trends in BCI-robotics for motor control and rehabilitation, *Current Opinion in Biomedical Engineering*, Vol. 20, 2021, 100354.

[2]. Muhammad Ahmed Khan, Rig Das, Helle K. Iversen, Sadasivan Puthusserypady, Review on motor imagery based BCI systems for upper limb post-stroke neurorehabilitation: From designing to application, *Computers in Biology and Medicine*, Vol. 123, 2020, 103843.

[3]. Xiaorong Gao, Yijun Wang, Xiaogang Chen, Shangkai Gao, Interface, interaction, and intelligence in generalized brain–computer interfaces, *Trends in Cognitive Sciences*, Vol. 25, Issue 8, 2021, pp. 671-684.

[4]. Juan A. Gallego, Tamar R. Makin, Samuel D. McDougle, Going beyond primary motor cortex to improve brain–computer interfaces, *Trends in Neurosciences*, Vol. 45, Issue 3, 2022, pp. 176-183.

[5]. Drishti Yadav, Shilpee Yadav, Karan Veer, A comprehensive assessment of Brain Computer Interfaces: Recent trends and challenges, *Journal of Neuroscience Methods*, Vol. 346, 2020, 108918.

[6]. Alexander E. Hramov, Vladimir A. Maksimenko, Alexander N. Pisarchik, Physical principles of brain–computer interfaces and their applications for rehabilitation, robotics and control of human brain states, *Physics Reports*, Vol. 918, 2021, pp. 1-133.

[7]. Nathan Dunkelberger, Eric M. Schearer, Marcia K. O'Malley, A review of methods for achieving upper limb movement following spinal cord injury through hybrid muscle stimulation and robotic assistance, *Experimental Neurology*, Vol. 328, 2020, 113274.

[8]. Yuancheng Li, Tianwei Zheng, Mei Wang, Lihong Dong, Pai Wang, Xuebin Qin, A coloring and timing brain-computer interface for the nursing bed robot, *Computers & Electrical Engineering*, Vol. 95, 2021, 107415.

[9]. Yizhi Liu, Mahmoud Habibnezhad, Houtan Jebelli, Brain-computer interface for hands-free teleoperation of construction robots, *Automation in Construction*, Vol. 123, 2021, 103523.

[10]. Nida Fatima, Ashfaq Shuaib, Maher Saqqur, Intra-cortical brain-machine interfaces for controlling upper-limb powered muscle and robotic systems in spinal cord injury, *Clinical Neurology and Neurosurgery*, Vol. 196, 2020, 106069.

[11]. D. Kuhner, L. D. J. Fiederer, J. Aldinger, F. Burget, M. Völker, R. T. Schirrmeister, C. Do, J. Boedecker, B. Nebel, T. Ball, W. Burgard, A service assistant combining autonomous robotics, flexible goal formulation, and deep-learning-based brain–computer interfacing, *Robotics and Autonomous Systems*, Vol. 116, 2019, pp. 98-113.

[12]. Robert Bauer, Meike Fels, Mathias Vukelić, Ulf Ziemann, Alireza Gharabaghi, Bridging the gap between motor imagery and motor execution with a brain–robot interface, *NeuroImage*, Vol. 108, 2015, pp. 319-327.

[13]. Gloria Beraldo, Stefano Tortora, Michele Benini, and Emanuele Menegatti. Hybrid brain-robot interface for telepresence, in *Proceedings of the workshop AIRO-2020 of the 19th International Conference of the Italian Association for Artificial Intelligence (AI*IA 2020)*, 2019, pp. 6-11.

[14]. Alkinoos Athanasiou, George Arfaras, Niki Pandria, Ioannis Xygonakis, Nicolas Foroglou, Alexander Astaras, and Panagiotis D. Bamidis1, Wireless brain-robot interface: user perception and performance assessment of spinal cord injury patients, *Wireless Communications and Mobile Computing*, Vol. 2017, 2986423, pp. 1-16.

[15]. Iáñez E., Furió M.C., Azorín J.M., Huizzi J.A., Fernández E., Brain-robot interface for controlling a remote robot arm. In: Mira, J., Ferrández, J.M., Álvarez, J.R., de la Paz, F., Toledo, F.J. (eds.) Bioinspired Applications in Artificial and Natural Computation. IWINAC 2009, *Lecture Notes in Computer Science*, Vol. 5602, Springer, Berlin, Heidelberg.