Deep-Learning Assisting Cerebral Palsy Patient Handgrip Task Translation

Fazrul Faiz Zakaria¹², Mohd Nazri Mohd Warip¹², Phaklen Ehkan¹², Muslim Mustapa¹², Mohd Zaizu Ilyas²

¹Advanced Computing, Centre of Excellence (CoE), Universiti Malaysia Perlis (UniMAP), Perlis, Malaysia.
²Faculty of Electronic Engineering Technology, Universiti Malaysia Perlis (UniMAP), Perlis, Malaysia.

ffaiz@unimap.edu.my

Abstract. An electro-encephalography (EEG) brain-computer interface (BCI) can provide the brain and external environment with separate information sharing and control networks. EEG impulses, though, come from many electrodes, which produce different characteristics, and how the electrodes and features to enhance classification efficiency have been chosen has become an urgent concern. This paper explores the deep convolutional neural network architecture (CNN) hyper-parameters with separating temporal and spatial filters without any pre-processing or artificial extraction processes. It selects the raw EEG signal of electrode pairs over the cortical area as hybrid samples. Our proposed deep-learning model outperforms other neural network models previously applied to this dataset in training time (~40%) and accuracy (~6%). Besides, considerations such as optimum order for EEG channels do not limit our model, and it is patient-invariant. The impact of network architecture on decoder output and training time is further discussed.

1. Introduction
Cerebral palsy is a condition that involves disability in motor functionality, which is triggered during the prenatal and neonatal period by brain injury. The extrapyramidal tract inside the human body is the one that smoothens the movement gesture and posture. However, as the extrapyramidal tract damage for a person with the cerebral palsy condition, they cannot usually move their muscle as envisioned. Most of the patients are diagnosed with tension-athetosis-type (TAT) cerebral palsy, where they are having difficulty maintaining a proper posture continuously due to muscle stress. Besides their lack of motor functionality, the TAT cerebral palsy patient brain activity is still intact, which leads to advancement in the Brain-computer interfaces (BCIs) from both industry and academia. The BCIs intended to assist TAT cerebral palsy individuals in their daily life activities. One can now control assistive devices utilizing the brain signals to enable them to be independent in daily life routine [1].

BCIs receive, interpret, and transform brain impulses into commands relayed to output devices performing desirable action. It is a process of encoding and decoding electrical brain activities. Encoding the electrical brain signal using a non-invasive methodology is the advantage of electro-encephalography (EEG). Researchers have gone on to use the EEG in encoding the electrical activity of the brain. Although this paper decoding algorithm using data acquired from scalp EEG data, More
intrusive brain signals, including intracortical, microelectrode (MAE), electrocorticographic (ECoG), and other brain signals, are also can be applied by this algorithm [1]–[5].

Achieving a high accuracy result in decoding complex brain signals from EEG is the main objective of most proposed decoding algorithm and architecture. Over the years, researchers have been presenting different techniques to reduce brain activity data's dimension to make it compatible with pattern recognition algorithm. The filter bank common spatial pattern (FBCSP) is one of the widely adopted decoding techniques [6]–[8]. FBCSP involves several phases of pre-processing, filtering, feature extraction, and selection before passed as input data for a classification algorithm. However, these FBCSP phases often involve a manual data interpolation, becoming a great challenge for BCIs to archive a full automation solution [9].

A significant effort on archiving automatic data interpolation is still ongoing. Researchers in the field had proposed several automated versions of the FBCSP algorithm, yet challenging to archive better accuracy [10], [11]. The deep-learning algorithm has recently gained popularity as a BCIs decoding technique. This neural-network-based algorithm can analyze a vast amount of data with higher accuracy than existing algorithms, especially in the field of computer vision [12], [13]. Two known deep-learning algorithms are convolutional neural networks (CNNs) and recurrent neural networks (RNNs), each suitable for different applications. This paper focuses on the CCNs family of deep-learning algorithms, which has transformed computer vision over the past decade. The advantage of CCNs in extracting classification features from data set automatically addressing the bottleneck in conventional machine learning [14]–[16].

The performance of CNNs depends heavily on the number of convolutional layers, affecting the overalls CNNs training time. This work presents a CNNs architecture exploration to evaluate the architecture parameters variation effect on classification algorithm performance to determine the handgrip activity from brain signals. The proposed CNNs algorithm outperforms the conventional machine-learning technique on the brain signals dataset while addressing the shortcoming of inter-subject variability. This paper is organized as follows: In section II, the experimental setup is presented. The proposed classification algorithm and architecture are presented in section III, and the techniques used in this design are discussed in section IV. In section V, implementation results are presented. Section VI concludes the paper.

2. Methods and Dataset
Scalp EEG data were collected from four (4) individuals (two (2) males and two (2) mean age of 20 ± 3.5 years) who performed a self-paced handgrip task. All four individuals were diagnosed with cerebral palsy and were currently attending community rehabilitation centers. Well-versed written consent was obtained on behalf of all individuals from their parent or guardian. None of these individuals had previous experience with any BCIs related devices. The session recorded per individual was speckled based on the therapy routine schedules during this experiment—only one individual is known as a right-handed user. In contrast, others are unknown due to their condition.

A 64 channel Quick-Cap (SynAmps2) and a 24-bit amplification device (SynAmps2) were used in our data collection setup to acquire EEG results. However, not all the electrodes were applied to all our participants. Using the same acquisition method, the electromyography (EMG) is concurrently capturing, and the movement time was time-locked to the EEG. The EMG surface electrodes were mounted on the forearms of the flexor digitor. A bandpass filter sampled the data at 1 kHz with a 0.1 to 100 Hz windowing mechanism filtered at 50 Hz. As introduced in CURRY 7, Hamming window was first used to register both EEG and EMG data and then filtered further using a zero-phase delay FIR (Hann window) filter.

With the physician's help, each individual was seated upright on a wheelchair and instructed to grasp their left and right hands alternately with open eyes and concentrate their eyes straight upfront. No intervention was prearranged to encourage the individuals to grip his/her hand to reduce the training data's effect evoked by the visual or auditory potential. Each participant was provided with a squeeze ball in each hand. Recordings were carried out in five-minute sessions, with gaps in between. Total data
per session was recorded for about 20-30 minutes. Each data from EEG were decimated to 250 Hz before we started training the network. Table 1 summarized the number of handgrips by each individual.

2.1. Data Pre-processing and Feature Extraction

The EEG data was marked into different classes by extracting the root mean square (RMS) power from the EMG to archive the motion recognition. At each sliding window of 500 ms, the RMS power was evaluated. A motion was observed as the RMS power value crossed a manually defined threshold. Before the experiment started, this subject-specific threshold was calculated by visual observation of the baseline RMS power of different handgrips. To ensure the recorded stage's consistency, we help the participants sustain identical hand-squeeze gestures and keep their arms steady between each movement.

Next, the EEG data were separated into epochs using EMG labels, 100 ms before the handgrip begins and 300 ms after the handgrip is released. A refractory duration of 300 ms was added after a grip was observed to prevent a single clench from being recorded twice. Period's in-between handgrip was often used as indicators of no-movement signals, i.e., occasions when the individual did not grip either hand for at least 500 ms. A similar window size of 400 ms was used for the no-movement EEG epochs. Table I shows the number of training examples obtained for the no-movement right-handgrip and left-handgrip on all four (4) individuals tested. As observed, the total of right and left epochs is approximately balanced, and so an unbiased classification model is more likely to be developed.

| Individual | EEG Channels | Total Right Grip Samples | Total left Grip Samples | Total Rest Samples |
|------------|--------------|--------------------------|-------------------------|-------------------|
| I          | 62           | 731                      | 317                     | 329               |
| II         | 31           | 657                      | 326                     | 321               |
| III        | 46           | 759                      | 546                     | 563               |
| IV         | 10           | 743                      | 507                     | 499               |

2.2. Classification Algorithm

Due to the three labels dataset's warped nature, we decided to limit the experiment to the binary representation of right vs. left grip actions. Besides, a two-dimensional matrix in the context of a binary classifier is sufficient to prove the methods. The data is then presented to the classifier, where the number of channels represents columns, and the rows represent the time.

CNNs are highly influenced by the biologically visual cortex's architecture and are a particular feed-forward neural network subtype. There are two sets of layers in CNNs: 1) The convolutions layers that translate the input matrix into different smaller vectors that function as classification features. 2) A fully connected layer that takes the output of convolution/pooling and predicts the best label to describe and perform the classification task on it.

The convolutional layers convolve a kernel window on the input with a certain measure to delete features from the input (step size). It is similar to local receptive fields in visual systems. Multiple kernels can be used by each convolution layer, leading to multiple feature maps produced out of the same input. The weights correlated with a kernel are constant for and feature map, making convolutions layers faster than regular feed-forward layers that need to learn various weights for different neuron groups.

Each successive convolution layer raises the number of kernels (and thus feature maps) relative to the previous layer but gains further high-level features resulting from combining previous neurons' output. The completely linked layers have all of their neurons connected to all of their previous layer's output. They conduct the classification and produce a probability vector that defines the probability that input is given with each classification label.

Figure 1 shows the proposed high-level architecture CNN model. There are two successive convolution layers induced by ReLU, with k and 2k neuron layers. Both of these layers performing input
signal feature extraction. A fully connected layer of m ReLU-activated neurons is used for classification, followed by a single sigmoid-activated neuron in the output layer.

The kernel window slides over the input with steps one (1) on the first and two (2) on the second layers, respectively. Compared to computer vision and image processing applications with a narrow kernel window, our window has width and height of the total EEG channels. We capture all channels' dependencies instead of collecting local dependencies between the input data and channels. This method is independent of the order of submission of channels to the classification model and does not depend on the channel's ideal sequence. To verify this, we performed multiple experiments where the channel sequence within the data input was modified. The results were not significantly varied.

Another distinction within this model is the CNNs defined by the absence of an image that aggregates a neuron group (using average or max) operation to minimize the model's size after use—then getting seen in intermediate convolution layers to diminish the model size. We assume this approach is only true when image recognition tasks do not greatly influence classification efficiency. To test this theory, we conducted experiments: the addition of a bundling layer effectively decreased the classification output consistently in pooling layers. The pool layers are sub-sample layers.

In the Keras system, we introduced and trained the model. The training procedure used was a stochastic gradient backpropagation with a decent batch size of 100 and a 1% constant learning rate over 90 epochs. Therefore, a random 100 input samples were chosen to trains each of the 90 training epochs. In all three neural network hidden layers, a 25% drop out [14] was used to prevent over-fitting. During training in the drop-out regularization phase, certain neurons in each layer are "shut down" arbitrarily, so the weights correlated with them have not altered in that specific period. The co-dependence of the neuron will be decreased and thus over-fits the training data.

2.3. **Testing Framework**

In this work, grid-search is used to find the optimal hyper-parameters of our classification algorithm, which results in the most accurate predictions. The hyper-parameter is a feature of the model's external model, the value of which cannot be estimated from data. Before the learning process starts, the value of the hyper-parameter has to be set.
The grid-search will explore the outcome of varying three key hyper-parameters in this work algorithm architecture.

1. The total number of convolutional kernel layers (k). The value varies between $2^2$ and $2^6$.
2. The total number of neurons (m) connected to the network is taken between 50 and 550, with 100 steps increments.
3. The kernel window width (w). Between 1 and 11 odd-numbered values were used.

Five-fold cross-validation is used for each hyper-parameter set to test the classifier. A randomly assigned hidden test-set (20 percent of the data) and a training set were part of each iteration of the five-fold cross-validation (80 percent of the data). During the CNN training phase, the test sets were not used. The classification efficiency was calculated in terms of accuracy for each test collection.

3. Results and discussion

The grid-search method was performed on each individual separately. While there were variations between individuals in the best overall hyper-parameters, one model obtained near-optimal outcomes for all individuals. Thus, the specification of the right model and hyperparameters was independent of the subject's heterogeneity.

3.1. Performance variation

The outcomes gap among the worst and best set of hyper-parameters was dissimilar for each person but ranged from 5% to 10%. (8 percent average). This indicates that it is important to choose the BCI decoder hyper-parameter architecture to gain better performance. It is important to remember that the optimal hyper-parameters set for each individual were not special and that several sets contributed to an optimal hyperparameter.

The set of hyper-parameters that produced the best outcome was different for each person, as described above. On the value of these parameters, however, there have been general developments:

- the smaller number of kernel nodes in the convolutional layers have contributed to improved results. In the first convolution layer, the optimal range for all users was eight kernels (and 16 in the next convolutional layer).
- The size change of the fully connected layer and the kernel window did not result in a major performance difference. However, increasing the number of neurons and glass diameter reduced the training time.

The outcome from this exploration we had selected n=8, w=1, and m=50 as the ideal hyper-parameter range for all individuals to construct a target-independent model. With 3% and 4% less precision than the best hyper-parameter mixture sets, this choice provided the best results for two persons and

---

Figure 2. Variations in the 3 investigated hyper-parameters (w, m and k) and their effect on classifier training time
suboptimal results for the other two. Table 3 explains the efficiency of the underlying algorithms with the best performance, the worst performance, and the selected hyperparameter sets.

3.2. Effects on training time
Although kernel window width on output and training time was marginal, their effects are significant. Figure 2 indicates the relationship between training time and these principles. Our selected parameters yielded the lowest training time and outperformed the best parameters set used previously.

Table 2. Best and Worst Hyper-parameter Set for Each Individual

| Individual | Best Set | Worst set |
|------------|----------|-----------|
| I          | w=5, m=50, k=8 | w=1, m=50, k=1288 |
| II         | w=1, m=50, k=8  | w=11, m=50, k=64  |
| III        | w=1, m=150, k=8 | w=11, m=50, k=128 |
| IV         | w=11, m=50, k=8 | w=7, m=50, k=128  |

3.3. Dataset evaluation on others model
To evaluate our proposed model's performance, we tested the same dataset on two neural network models. The MLP models presented in [15] have been shown to work the best. In that analysis, using a genetic algorithm, a separate shallow feed-forward network was built for each individual.

Just one individual (IV) was tested for the CNN in [12], and the CNN behaved the same as the MLP model. However, the kernel window's limited size created a problem since the channel order was far from optimum. Due to the greater neuron counts across all layers, the CNN needed more training time and larger models than our CNN. Our single one outperformed the model for all the variables included. The efficiency of the method is summarized in Table 4.

Table 3. Classification Accuracy on Different Hyper-parameters Sets

| Individual | Best Set Accuracy | Worst set Accuracy | Chosen case accuracy | Difference between best and worst | Difference between best and chosen |
|------------|------------------|--------------------|----------------------|----------------------------------|-----------------------------------|
| I          | 81%              | 71%                | 77%                  | 10%                              | 4%                                |
| II         | 87%              | 78%                | 87%                  | 9%                               | 0%                                |
| III        | 90%              | 85%                | 90%                  | 5%                               | 0%                                |
| IV         | 84%              | 76%                | 81%                  | 8%                               | 3%                                |

Table 4. Performance of Proposed Deep-learning vs. Existing Machine Learning

| Individual | Proposed DL Performance | MLP Performance |
|------------|-------------------------|----------------|
| I          | 75%                     | 79%            |
| II         | 80%                     | 86%            |
| III        | 80%                     | 88%            |
| IV         | 75%                     | 83%            |
4. Conclusion
Via our CNN platform, our team implemented our work on an established hand-squeeze electroencephalogram (EEG) dataset, which outperforms other deep learning models. The model's design did not rely on the order in the channels were introduced to it. The model needed less preparation in comparison to other models. The results were obtained by selecting the kernel window and optimizing hyperparameter settings at each iteration during grid search.

The model learned dependencies across channels by using a kernel window covering the whole EEG channel array, which was not feasible in previous models that used a smaller window.

To refine the architecture, we used grid check to evaluate differences in the number of kernels in the convolutional layer, the number of neurons in the completely connected layer, and the kernel's width in the convolutional layers. We showed that it is possible to tune those analytical parameters to contribute to all participants' nearly optimal results.

Even if the participants interact through the same networking architecture, they are independently trained over their respective data sets. The next portion of the report would discuss the disadvantages of the present analysis. They used electrodes that can be utilized together to test models dependent on both patients. Via transfer learning, we will build a pre-trained model that uses data from the whole set.

5. Acknowledgment
The author gratefully acknowledges financial support from the Malaysian Ministry of Higher Education (MOHE) under the Fundamental Research Grant Scheme (FRGS) (Grant number: FRGS/1/2018/ICT03/UNIMAP/02/3).

References
[1] R. O. Heidrich, E. Jensen, F. Rebelo, and T. Oliveira, “A comparative study: Use of a Brain-computer Interface (BCI) device by people with cerebral palsy in interaction with computers,” An. Acad. Bras. Cienc., 2015.

[2] S. Vaid, P. Singh, and C. Kaur, “EEG signal analysis for BCI interface: A review,” in International Conference on Advanced Computing and Communication Technologies, ACCT, 2015.

[3] R. Thota, S. Vaswani, A. Kale, and N. Vydyanathan, Machine Intelligence and Signal Processing, 2016.

[4] S. Aggarwal and N. Chugh, “Signal processing techniques for motor imagery brain computer interface: A review,” Array, 2019.

[5] M. Agustina Garcés and L. L. Orosco, “EEG signal processing in brain-computer interface,” in Smart Wheelchairs and Brain-computer Interfaces: Mobile Assistive Technologies, 2018.

[6] S. H. Park, D. Lee, and S. G. Lee, “Filter Bank Regularized Common Spatial Pattern Ensemble for Small Sample Motor Imagery Classification,” IEEE Trans. Neural Syst. Rehabil. Eng., 2018.

[7] S. C. Liao, C. Te Wu, H. C. Huang, W. T. Cheng, and Y. H. Liu, “Major depression detection from EEG signals using kernel eigen-filter-bank common spatial patterns,” Sensors (Switzerland), 2017.

[8] V. Gandhi, G. Prasad, D. Coyle, L. Behera, and T. M. McGinnity, “Quantum neural network-based EEG filtering for a brain-computer interface,” IEEE Trans. Neural Networks Learn. Syst., 2014.

[9] A. S. Aghaie, M. S. Mahanta, and K. N. Plataniotis, “Separable common spatio-spectral patterns for motor imagery BCI systems,” IEEE Trans. Biomed. Eng., 2016.

[10] M. M. H. Joy et al., “Multiclass mi-task classification using logistic regression and filter bank common spatial patterns,” in Communications in Computer and Information Science, 2020.

[11] A. Schwarz, R. Scherer, D. Steyrl, J. Faller, and G. R. Muller-Putz, “A co-adaptive sensory motor rhythms Brain-Computer Interface based on common spatial patterns and Random Forest,” in Proceedings of the Annual International Conference of the IEEE Engineering in
[12] V. J. Lawhern, A. J. Solon, N. R. Waytowich, S. M. Gordon, C. P. Hung, and B. J. Lance, “EEGNet: A compact convolutional neural network for EEG-based brain-computer interfaces,” *J. Neural Eng.*, 2018.

[13] H. Cecotti and A. Gräser, “Convolutional neural networks for P300 detection with application to brain-computer interfaces,” *IEEE Trans. Pattern Anal. Mach. Intell.*, 2011.

[14] A. Khan, A. Sohail, U. Zahoor, and A. S. Qureshi, “A survey of the recent architectures of deep convolutional neural networks,” *Artif. Intell. Rev.*, 2020.

[15] K. Abdelouahab, M. Pelcat, J. Sérot, and F. Berry, “Accelerating CNN inference on FPGAs: A survey,” *arXiv*. 2018.

[16] X. Zhang, C. Xv, M. Shen, X. He, and W. Du, “Survey of Convolutional Neural Network,” 2018.