Neural Network Sparsity in Brain-Body-Machine Interfaces

Laura C. Petrich[1,4,*], Samuel Neumann[1,4,*], Patrick M. Pilarski[1,2,4], and Alona Fyshe[1,3,4]

***Abstract*— Brain-body-machine interfaces acquire, process, and translate brain signals for individuals with severe motor impairments to communicate and control the assistive technology that supports their daily life activities. Electroencephalography (EEG) is a standard approach for acquiring such brain signals due to its low cost and high temporal resolution. EEG signals can be thought of as a proxy for the user's intent. One established method for translating this intent into inferences and actions are neural networks. However, densely connected neural networks can be computationally expensive—a problem for real-time, deployed brain-body-machine interface systems. In this paper we investigate the use of *sparsity* in neural networks for EEG-based motor classification, with the goal of reducing the number of neuronal connections without sacrificing a system's performance. We compare two sparsity-inducing algorithms, weight pruning and sparse evolutionary training, with a dense neural network under three experimental conditions. Overall, our results show that sparse neural networks can achieve higher performance accuracy and generalization than their densely-connected counterparts for an EEG-based classification task. We found that sparse evolutionary training achieves the highest and most stable performance across all experiments. Introducing sparsity into the network is a viable option for efficient EEG-based control, with promising applications in a range of related rehabilitation and assistive technologies. This brings us closer to helping individuals with severe motor impairments reclaim independence through more computationally realizable methods of interacting with their technology and the world around them.**

I. Introduction

Brain-body-machine interfaces (BBMIs) are an emerging technology aimed at enabling the control of external devices via acquired brain signals and other signals from the human body [1]. BBMIs have improved the quality of life for individuals with physical disabilities by increasing their personal autonomy [2]. The overarching goal of a BBMI system is to offer an alternative means of communication in order for the user to interact with their environment. A BBMI system is composed of three components: signal acquisition, signal processing (involving feature extraction, classification, and translation), and the control application [1]. The overall performance and success of a BBMI relies on the quality of signals acquired from the brain or body as these are used to interpret the user's intent. In this work we compare the performance of three neural network training regimes, shown in Fig. 1, on a BBMI task.

This work was supported by the Alberta Machine Intelligence Institute (Amii), the National Science and Engineering Research Council (NSERC), Alberta Innovates, and in part through the Canada CIFAR AI Chairs Program. [1]Dept. of Computing Science, [2]Dept. of Medicine, and [3]Dept. of Psychology, University of Alberta, Edmonton, AB, Canada; [4]Alberta Machine Intelligence Institute (Amii), Edmonton, AB, Canada.

* These authors contributed equally to this work. Please direct all correspondence to laurapetrich@ualberta.ca

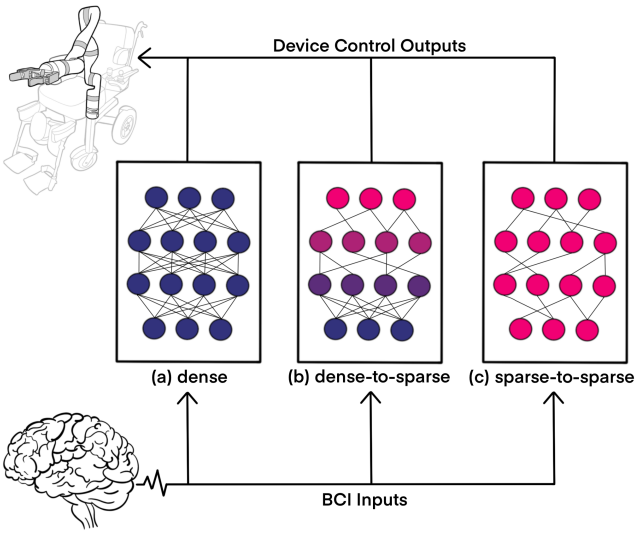


Fig. 1: Sparsity can be dynamically introduced into (a) *densely* connected neural networks by either starting with a dense layer and pruning network connections, known as (b) *dense-to-sparse* training, or initializing a sparse topology and iteratively pruning and regrowing connections, i.e., (c) *sparse-to-sparse* training. In this work we compare the performance of these three training regimes on a brain-computer interface task mapping electroencephalography signals (BBMI inputs) to upper limb motion events (device control outputs).

Electroencephalography (EEG) is a method of capturing the brain's electrical activity (i.e., signal acquisition) through wet or dry electrodes placed on the scalp [3]. It is popular for real-time control as EEG systems are relatively low-cost, non-invasive, and portable [4]. EEG signals have a high temporal resolution, however, they are inherently noisy and have poor spatial resolution [5]. Signal processing extracts a clean signal from a noisy EEG recording and then generates a mapping to the user's intent [6]. First, a preprocessing step is performed, involving channel selection (i.e., determine features relevant to the task of interest), signal frequency filtering (i.e., what filters should we apply to remove noise), and artifact removal (i.e., identify and remove irrelevant noise artifacts) [7]. Once the relevant features are extracted they can be classified and translated into motor commands to send to an external device (i.e., the control application), e.g., to control a smart wheelchair or robotic arm [1].

Due to the high-dimensionality, non-stationarity, and complexity of EEG data, dense neural networks are often used

for classification [8]. However, these networks can be computationally expensive and have large memory requirements. For practical control with EEG-based BBMI systems, low memory and computation requirements are necessary. The deployed system will need to be portable for improved usability, particularly for applications that do not have a local computer or internet connection. BBMI systems should be able to predict the user's intent accurately and react quickly to a user's input or feedback. Sparse neural networks are an emerging research area in machine learning that address both of these constraints [9], making them particularly well suited for BBMI control.

In a sparse neural network, each neuron is connected to a subset of neurons in the proceeding layer. This is in contrast to dense neural networks where each neuron is connected to all of the neurons in the proceeding layer. Sparse neural networks can be more energy efficient than dense neural networks, using a fraction of the memory and compute while reaching higher levels of performance and generalization [9]–[12]. In a recent survey on sparsity in deep learning, Hoefler et al. [13] found "that today's sparsification methods can lead to a 10-100x reduction in model size, and to corresponding theoretical gains in computational, storage, and energy efficiency, all without significant loss of accuracy" (p. 2–3). Furthermore, sparse neural networks may be more resistant to interference and catastrophic forgetting than their fully connected counterparts [14]–[16]. Finally, dense neural networks generally require large amounts of training data to generalize well, an important consideration in the application of BBMI systems where training datasets are usually small [8].

In this work *we investigate how sparsity affects the performance and generalization capabilities of neural networks in a classification task with EEG data.* Our key contribution is an empirical comparison of two techniques for adding sparsity to neural networks used for EEG-based classification, specifically examining their performance in the presence of variability both within individual participants and across different participants. We hypothesize that, when training and testing on a fixed pool of participants, with fewer neuronal connections sparse neural networks can attain higher levels of prediction accuracy and generalization compared to a dense neural network.

II. Sparsity in neural networks

Sparsity can be introduced into neural networks either *statically* or *dynamically*. Static sparse networks have fixed sparse connections between neurons and are not adapted during training [17]. Sparsity in these networks is due to deliberate architecture choices. In contrast, dynamic sparse networks have sparse connections that are learned through a gradual prune-and-regrow procedure [13]; it is this class of sparse networks we consider in the present study, due to their experimentally suggested utility within the animal brain. For example, Barth and Poulet [18] provide empirical evidence that the brain exhibits sparse firing of neocortical neurons, allowing for efficient representation of sensory information.

Algorithm 1: Sparse Evolutionary Training

Input: prune fraction ζ, prune frequency n, sparsity level ε, optimiser $\mathcal{O}$, loss $\mathscr{L}$
Initialize network with Erdős-Rényi topology with sparsity level ε
for *each training epoch* e **do**
  Train network parameters using $\mathcal{O}$ on $\mathscr{L}$
  if $e \equiv 0 \pmod{n}$ **then**
    Remove $\frac{100\zeta\%}{2}$ smallest positive weights
    Remove $\frac{100\zeta\%}{2}$ largest negative weights
    if e *is not the last epoch* **then**
      Add random new connections (weights) between consecutive layers equal to the amount removed
    end
  end
end

Algorithm 2: Weight Pruning

Input: prune fraction ε, prune epoch n, optimiser $\mathcal{O}$, loss $\mathscr{L}$, final network topology $\mathcal{T}$
for *each training epoch* e **do**
  Train network parameters using $\mathcal{O}$ on $\mathscr{L}$
  if $e = n$ **then**
    Remove $100\varepsilon\%$ of weights of lowest magnitude, ensuring that the resulting network has topology $\mathcal{T}$
  end
end

Dynamic sparse networks have also been argued to be more biologically plausible than static sparse networks, given the way that the animal brain is known to dynamically adjust its neuronal connections [19]–[21].

Dynamic Sparse Training (DST) refers to the simultaneous optimization of a dynamic sparse network's weights and neuronal connections. The Sparse Evolutionary Training (SET) algorithm of Mocanu et al., shown in Algorithm 1, is one of the seminal works in the DST line of research [10]. SET follows a *sparse-to-sparse* training regime (e.g., Fig. 1c), meaning that it initializes a neural network with a sparse Erdős-Rényi random graph topology [22] with sparsity level $\varepsilon \in (0, 1)$ and subsequently learns the weights and connections. Every n epochs, SET prunes and re-grows the connections. To do so, it first removes the $\frac{100\zeta\%}{2}$ smallest positive and largest negative weights (i.e., weights closest to zero) for $\zeta \in (0, 1)$. Next, an equal number of random connections are added back into the network, preserving the Erdős-Rényi topology. Conventional training then continues until the next pruning epoch. Mocanu et al. [10] show that networks trained with SET can match the performance of densely connected networks with a fraction of the number of neurons and far less memory and computation.

In contrast, *dense-to-sparse* training algorithms (e.g., Fig. 1b) begin by initializing a dense network then introduce sparsity by pruning network connections during training [23]–[26]. Such methods generally follow a three stage process. In the first stage, a densely connected network is

trained following conventional training procedures. During this stage, the learned network weights are considered to represent the importance of each connection where the importance of a weight is measured by its magnitude. In the second stage, the $100\varepsilon\%$ of unimportant connections are pruned at epoch n for $\varepsilon \in (0, 1)$. Finally, the pruned, sparse network is re-trained from either the current weight values or after re-initializing the weights. A typical weight pruning algorithm (WP) adapted from Han et al. [24], and the one we consider in this work, is shown in Algorithm 2.

III. Methods

A. Data characteristics and considerations

For this study work with the upper-limb prosthesis WAY-EEG-GAL dataset collected by Luciw et al. [27]. This dataset is well established in the community and has been used in classification tasks to achieve a high classification accuracy [28]–[30]. It was gathered with the intent of evaluating whether EEG signals can be used for upper-limb prosthetic device control. The data was collected with twelve right-handed participants and contains a total of 3,936 grasp and lift trials (328 trials per participant). Each participant completed ten series consisting of approximately 30 trials within each series. For each trial the participant reached out, grasped an object, lifted the object, and held it in the air for a few seconds before placing it back down and releasing the object. A hand motion event label was assigned to an EEG recording by Luciw et al. [27] using a combination of velocity and force data to determine the event onset. A subset of the original dataset was then extracted for a Kaggle competition where an event label was also assigned to an EEG sample if it was within 150 ms of the event onset [31]; it is this subset that we used for our study. For detailed information on the data collection protocol we refer the reader to the original article by Luciw et al. [27].

Since the events occur sequentially (i.e., time-series data), each EEG sample may be labelled with multiple events. In order to better isolate the effect of introducing sparsity, we extracted four event classes that did not have overlapping event labels for training our classifiers: HandStart (class 0), LiftOff (class 1), Replace (class 2), and BothReleased (class 3). The class label, class number, and number of samples for each class are summarized in Table I. Each sample (i.e., model input) contains 32 channels of raw EEG data from wet electrodes collected at a 500 Hz sampling rate with a corresponding event label (i.e., there is a signal-label sample every 0.002 seconds).

TABLE I: Class details for the dataset used in our multi-class classification problem. A weighted cross-entropy loss function was used to help address the class imbalance.

Class Label	Class Number	Number of Samples
HandStart	0	468000
LiftOff	1	388279
Replace	2	321905
BothReleased	3	321905

B. Data pre-processing

EEG data is known to be noisy and requires filtering in order to be sure that artifacts are not interfering with model predictions [32]. Head or wire movements as well as perspiration can cause low frequency noise artifacts. High frequency artifacts can be caused by muscle contractions in the vicinity of the EEG recording sites (electromyography artifacts), eye blinking or movement (electrooculography artifacts), and electromagnetic interference. During data collection for the original WAY-EEG-GAL dataset, no pre-processing (e.g., artifact rejection) steps were applied to the EEG signals other than the removal of "unneeded or extra samples at the beginning and end of each series" (p. 5) [27]. In line with standard practices, e.g., Janapati et al. [33], we applied a bandpass filter with a low frequency cutoff of 0.2 Hz and a high frequency cutoff of 50 Hz. The frequency cutoff of 0.2 Hz will remove low frequency movement artifacts, while the 50 Hz cutoff will remove high frequency noise artifacts from muscle contractions, eye movements, and electromagnetic interference.

C. Algorithms and hyperparameter tuning

To better understand how sparse neural networks can perform on an EEG-based classification task, we studied the learning process of two dynamic sparse training (DST) algorithms. The first algorithm is sparse evolutionary training (SET), shown in Algorithm 1. The second is a weight pruning (WP) algorithm, shown in Algorithm 2, similar to those used by Frankle and Carbin [26] and Han et al. [24]. Following the advice of Han et al. [24] who found that retraining pruned layers from the surviving parameters resulted in better performance; we did not re-initialize weights after pruning, rather training continued from the current weight values.

We compared these two DST algorithms with two baselines. The first baseline is a small densely connected network, which we denote as DN. The second baseline randomly predicted class labels according to their relative occurrence frequency in the dataset. This baseline served as a sanity check to ensure that each algorithm learned reasonable behaviour. All algorithms were implemented in `Flax` [34] using the `jaxpruner` library [35].

Due to the nature of our study, we were not interested in attaining state-of-the-art performance, rather, we wanted to characterize the learning processes of sparsity-inducing algorithms. For all experiments to remain interpretable, all neural networks were composed of two hidden layers. We performed a sweep of hidden layer sizes in $\{128, 256, 512, 1024\}$ and found that all networks performed best with hidden layers of size 1024 and that the computation time for all network sizes was similar. We constrained ourselves to hidden layers of 1024 and below to keep our study focused on relatively small computational loads that are deployable on embedded systems. For the rest of the experiments we set the hidden layers to size 1024. Each network predicted a softmax distribution over class labels. All neural networks were trained using the Adam

optimizer [36] from the `Optax` library [37] with default hyperparameters, except for the stepsize.

To tune the stepsize and sparsity specific hyperparameters, we conducted a grid search for multiple runs with three random seeds for Experiments 1 and 2, and fifteen random seeds for Experiment 3. We selected the tuned hyperparameters based on the highest average prediction accuracy achieved throughout training. For all experiments we swept stepsizes in $\{0.01, 0.001, 0.0001\}$. In our preliminary experiments, we evaluated the prediction accuracy of SET and WP for sparsity levels $\varepsilon \in \{0.25, 0.5, 0.75\}$ and percentage of dropped weights (SET) $\zeta \in \{0.15, 0.3, 0.5\}$. For both the sparsity level and percentage of dropped weights, we found that prediction accuracy was highest for the lowest two values and swept over those values for the final experiments. We further performed a sweep of batch sizes and found that full-batch updating resulted in highest prediction accuracy across all algorithms. However, in a number of experiments, full-batch updating was not possible due to the immense size of the training data. We therefore used a large batch size of 8192. We tested pruning weights every $\{1, 2, 5\}$ epochs for SET (since this algorithm prunes many times during training) and once at epoch $\{167, 333, 400\}$ for WP (i.e., pruning once $\frac{1}{3}$ $\frac{2}{3}$, or $\frac{4}{5}$ of the way through the total number of epochs).

For all experiments we used a weighted cross-entropy loss to help address the class imbalance, similar to that in the `scikit-learn` library [38]. The loss term for each class y was weighted by $\frac{N}{Cn_y}$, where N is the total number of data samples in the dataset, C is the number of classes, and n_y is the number of data points with label y. We used a stratified 80/10/10 train, test, and validation split for each experiment.

D. Experiment details

We conducted three different experiments to test the efficacy of sparsity in different settings. First, each algorithm was evaluated when training and testing data came from the same pool of participants. Next, we evaluated how each algorithm performed when classifying out-of-distribution data, where the training and testing datasets came from different pools of participants. Finally, we investigated how increasing amounts of training data from a single participant affected each algorithm's predictions on the same participant.

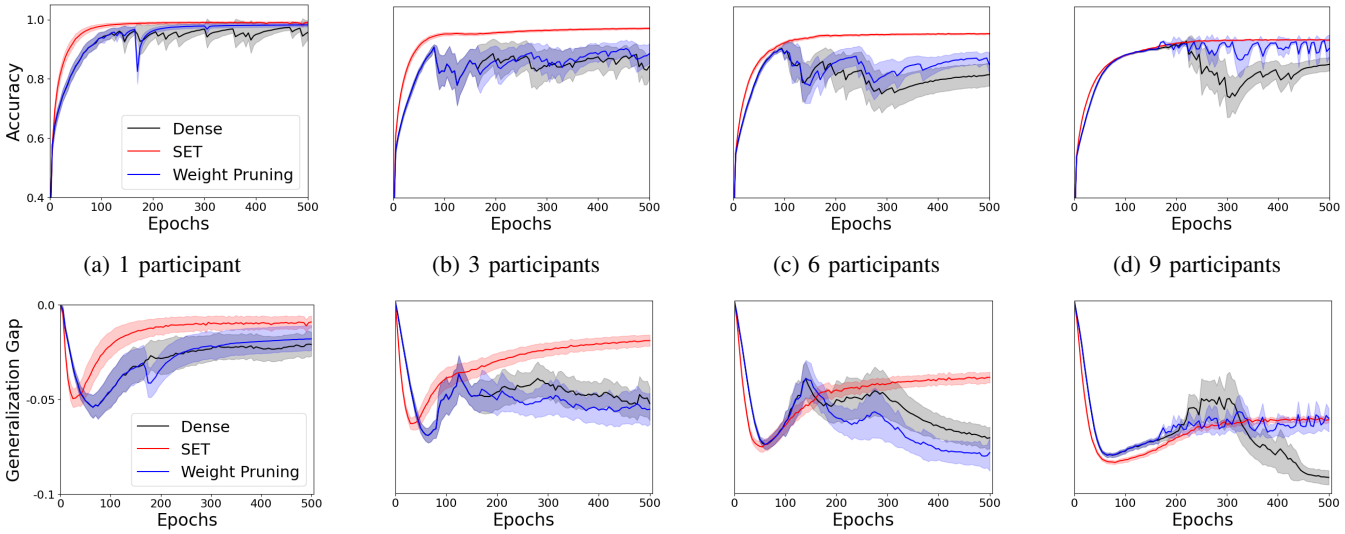
a) Experiment 1: Our first experiment addressed the question *how does each model perform within the participant pool it was trained on?* To answer this question, we trained and tested each algorithm on data from $N \in \{1, 3, 6, 9\}$ participants out of the first ten participants. The N participants were randomly sampled without replacement. We tuned hyperparameters over three random seeds. We then conducted an additional 20 experimental runs using the tuned hyperparameters and report performance metrics for these final 20 independent runs.

Due to the nature of the EEG data, we expected that increasing the number of participants in the training dataset would decrease prediction accuracy. This is because the model needs to account not only for variability that exists within an individual participant but also for variability between different participants. We were interested to see whether sparsity can help to counteract this effect. Indeed, for many of the highest performing submissions in the Kaggle competition, a separate network was trained for each participant [31]. Nevertheless, a BBMI system should be capable of being trained on multiple participants data in order to reduce the amount of training data required per individual.

b) Experiment 2: Our second experiment addressed the question *can each model efficiently generalize to the data of new, held-out participants?* For this experiment we took the tuned-and-trained models from Experiment 1 and tested how they performed on data from the two held-out participants 11 and 12. We expected each model to perform worse in this setting compared to Experiment 1, but we were interested in determining if sparsity helps to negate it (e.g., via improved generalization or reduced loss in prediction accuracy). This study allowed us to better understand how training data should be collected for real-world BBMI systems—from the end user or other participants. Improving generalization to new participants would be of great benefit to BBMI systems, as it is often challenging to have the end-user come in for multiple training sessions.

c) Experiment 3: Our final experiment sought to address the question *how does each algorithm perform with differing levels of training data for a single participant?* This experiment allowed us to better understand how each algorithm may behave in a real-world setting where the system could potentially learn and adapt to the behaviour of the end-user during deployment. For this experiment, data from a single randomly sampled participant from the first ten participants was split into training, validation, and test sets. The training set was composed of $p \in \{10\%, 20\%, 30\%, 50\%, 70\%, 80\%\}$ of the participant's data. The validation set was composed of 10% of the training set, and the test set was composed of the remaining data. Note that when the training data reaches 80% of the data this is equivalent to the 1 participant setting in Experiment 1. An increasing amount of training data should improve the performance of each algorithm, with diminishing returns. A well-performing algorithm, especially one suited for a deployable BBMI system, should reach reasonable performance with a moderate amount of training data. We repeated this experiment 15 times with different random seeds, tuning hyperparameters and reporting performance over all 15 seeds.

d) Evaluation metrics: Model performance was evaluated by prediction accuracy. Similar to Novak et al. [39] and Liu et al. [40], generalization is quantified as the *generalization gap*—the difference in prediction accuracy between the testing and training datasets. The generalization gap is expected to be less than zero, as it is unlikely that the test accuracy is greater or equal to the training accuracy. If a model were to perfectly generalize to unseen data, the generalization gap would be zero. We report mean evaluation metrics across runs with bootstrap confidence intervals. We used 10k resamples, batch sizes of 10, and 5% significance to construct confidence intervals, unless otherwise specified.



(a) 1 participant (b) 3 participants (c) 6 participants (d) 9 participants

(e) 1 participant (f) 3 participants (g) 6 participants (h) 9 participants

Fig. 2: Sparse Evolutionary Training (SET) displays consistently high accuracy and better generalization than other algorithms in Experiment 1, where both train and test data are from the same pool of participants. **Top Row**: Test accuracy over 20 independent runs. As more participants are added to the training dataset, the performance of SET stays consistently high with low variance. **Bottom Row**: Generalization gap (i.e., difference between the test and train accuracy) over 20 independent runs. SET is able to generalize more efficiently than either the dense network (DN) or weight pruning (WP). **Both**: Solid lines denote mean performance with shaded regions denoting 95% bootstrapped confidence intervals.

IV. Results

A. Experiment 1: Pooling participant's data

The top row of Fig. 2 shows the test accuracy for each algorithm over 20 independent runs with training/test data from 1, 3, 6, and 9 randomly sampled participants. WP performed similarly to DN in terms of prediction accuracy. In contrast, SET generally improved prediction accuracy compared to DN and learned noticeably faster than both DN and WP. Accuracy decreased for all algorithms when more participants were added into the training data. We hypothesize that this is because each algorithm had to better generalize among all participant's conditional data distributions.[1]

Whereas SET exhibited low variance in prediction accuracy with tight confidence intervals, DN and WP exhibited severe degradation in prediction accuracy at differing intervals throughout training as seen in the top row of Fig. 2. Notably, these dips are not apparent when using full-batch updating and happen across all classes (that is, the dips are not due to poor prediction accuracy for a single class). Such a phenomenon reduces the utility of both these algorithms to real-life BBMI systems.

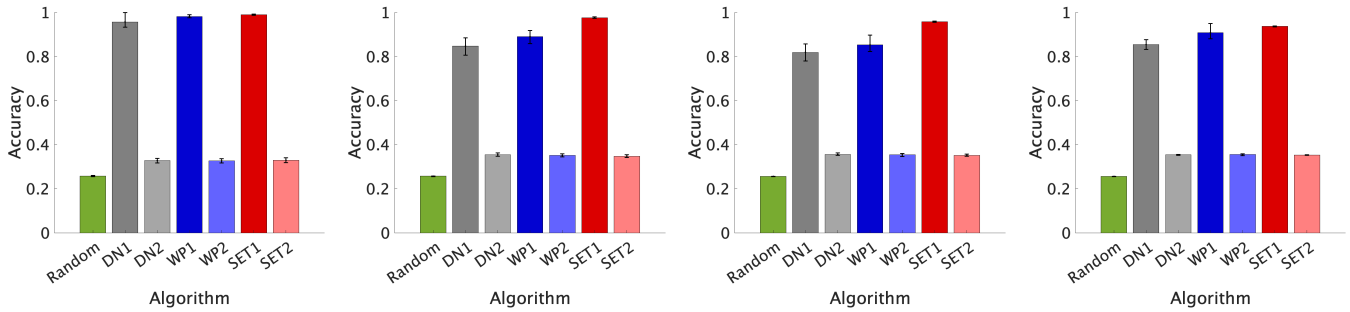
The bottom row of Fig. 2 outlines the generalization gap for each algorithm. SET often exhibited a significantly lower magnitude generalization gap than either DN or WP (note where the confidence intervals do not overlap). WP and DN exhibited a similar-magnitude generalization gap, except when training on 9 participants. For all algorithms, the generalization gap increased with an increasing number of participants. Overall, SET obtains the best prediction accuracy and performance stability across all settings suggesting that it is a helpful training regime when adding more participants data into the training pool.

B. Experiment 2: Generalizing to held-out participants

For this experiment we took the tuned-and-trained models from Experiment 1 and tested how well they performed on data from held-out participants 11 and 12. Fig. 3 shows the test accuracy of random predictions, DN, WP, and SET over 20 random seeds for both Experiment 1 and Experiment 2. Overall, we see a steep decline in accuracy when making predictions on held-out participant data, with algorithms performing only marginally better than the random baseline. For the 1-participant case, we see a $\sim 66\%$ decrease in prediction accuracy for all algorithms (comparing the Experiment 1 bars with corresponding Experiment 2 bars in Fig. 3).

Notably, this decline in prediction accuracy is not due to the held-out participants having poor quality data. We verified that each algorithm could learn on these participants' data, each reaching over 90% prediction accuracy. Overall, DN, SET, and WP each performed approximately equally when predicting on held-out participant data, with a slight increase in accuracy moving from the 1 to 3 participant setting (approximately 32% to 35%). This is unsurprising as EEG signals have been used for person recognition systems, suggesting there is sufficient variation in the signals between different people for classification. Considering these results in tandem with those of Experiment 1, we conclude that it

[1]Denote the theoretical distribution of EEG data (X) and labels (y) on this grasp-and-lift task as $p(X, y)$. Then the distribution of EEG data from a single participant S is $p(X, y \mid S)$.



(a) 1 participant (b) 3 participants (c) 6 participants (d) 9 participants

Fig. 3: The deep coloured bars (DN1, WP1, and SET1) show the Experiment 1 results while the faded bars (DN2, WP2, and SET2) show the Experiment 2 results. All network architectures show a steep decline in accuracy from the Experiment 1 results when making predicts based on out-of-distribution data from held-out participants. Bars show the accuracy of random predictions, dense network (DN1/DN2), weight pruning (WP1/WP2), and sparse evolutionary training (SET1/SET2) algorithms for Experiments 1 and 2, respectively, over 20 random seeds. Error bars denote 95% bootstrapped confidence intervals with 10,000 resamples.

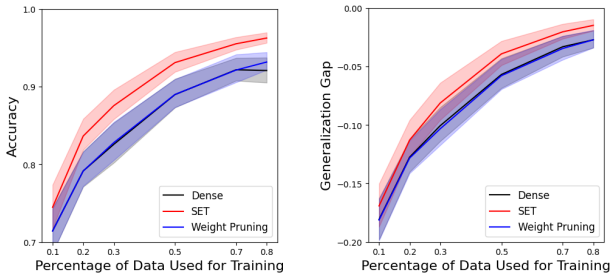
might be beneficial to include *data from the end-user when training a BBMI control system in order to help reduce the loss in model performance.*

C. Experiment 3: Single-participant models

A successful BBMI system should be able to continually learn and adapt to the dynamic nature of its user. This experiment allowed us to quantify the gains in performance with increasing data from the end-user. Ideally, a BBMI system would be useful even with a low amount of training data, although what defines a model as *useful* will ultimately depend on the end application.

Fig. 4a shows the accuracy during learning for increasing amounts of training data. Even with a low amount of data, all algorithms attained reasonable performance. With just 10% of the training data (approximately 27 grasp and lift trials), all algorithms reached a prediction accuracy around 75%. As more training data was introduced, the accuracy of all algorithms continued to increase, as expected. The prediction accuracy of SET increased more quickly than that of DN and WP and was significantly higher for datasets composed of more than 10% of a participant's data (note how the accuracy confidence intervals do not overlap). For the largest dataset sizes, SET attained significantly higher levels of precision and recall than DN (with a significance level of 0.1); table omitted for concision.

Fig. 4b shows the generalization gap for DN, SET, and WP. Overall, the gap shrinks with increasing training data, indicating that all algorithms can better generalize with more training data. When 10% of the data was used for training, the generalization gap was around 20% across algorithms—indicating overfitting. When more than 70% of the data was used for training, the generalization gap was less than 5%, indicating that the algorithms could generalize to unseen data effectively. All three algorithms exhibited a similar generalization gap for increasing dataset sizes.



(a) (b)

Fig. 4: The amount of participant data used in training impacts both accuracy and generalization, as shown via Experiment 3 results with train and test data from a single participant. Prediction (a) accuracy and (b) generalization gap when trained on increasing amounts of training data. Solid lines denote mean performance with shaded regions denoting 95% bootstrap confidence intervals. (a) SET attained the highest accuracy for increasing dataset sizes; (b) each algorithm exhibited a similar generalization gap.

V. Discussion

An open question in the BBMI community is *from whom should data be collected* in order to reduce the effects of intra- and inter-participant variability when training a BBMI control model [41]. We showed how the algorithms under consideration were able to make accurate predictions for participant data on which each algorithm was trained. But, when testing on held-out participant data, each algorithm exhibited a significant, and considerably large, drop in prediction accuracy, precision, and recall. This suggests that such computational models should include some training data from the end-user. Pre-trained models should be either fine-tuned to the end-user prior to deployment, for example through transfer learning [42], or include data from the end-user while training the model. For best performance, we suggest training a model on as much data as possible from solely the end-user, although we do acknowledge this may

not always be a viable option. When training on data from multiple participants, SET could help to minimize the decline in prediction accuracy.

Overall, our empirical study suggests that sparsifying a neural network is a viable option for reducing the number of neuronal connections when training an EEG-based model without significant loss of performance. As suggested by Lasby et al. [9], selective reduction may support the active alignment of network architectures to deployed computational hardware. Our study also indicates that sparse neural networks have the ability to adapt more quickly during training than their densely connected counterparts. This brings us one step closer towards making a *direct impact on the lives of people living with severe motor impairments through the use of machine learning models trained on brain data.*

A. Ethical considerations

As machine learning models become more prevalent in every day life, they are being used to make decisions that affect not only individuals but also society as a whole. Thus, it is important to understand the potential sources of harm that could arise during different stages of the machine learning life cycle. Suresh and Guttag [43] present a framework for identifying seven sources of harm, three of which we take into consideration for this study. The first is *representation bias*, which we note in the participant pool recruitment of the WAY-GAL-EEG dataset [27]; the population that would end up requiring and using a model such as this for BBMI control is underrepresented and models trained on this set could fail to generalize well for an assistive rehabilitation technology user population. The second is *measurement bias*. By oversimplifying a complex motor action recorded via EEG signals into four simple labels, we run the risk of a trained model being a poor reflection of user intent as measured from the whole brain. The third is *deployment bias*. A key goal of this work is to develop efficient and accurate prediction models for use in a real-world BBMI control system. Such a system would involve a human user controlling a neuroprosthetic device via real-time EEG signals. Thus, it is important that future work consider the match between the prediction problem that the model solves and how it is being used during deployment. Reproducibility and transparency are also an important part of ethical research practices. To this end, we open-sourced all of our code for dataset loading, model training, and evaluation.[2]

B. Limitations and future work

Our empirical analysis contained factors that limit the immediate transferability of our findings to deployed environments. One choice made in our data pre-processing step was to remove all samples with overlapping or no event labels. While making evaluation more straightforward, this choice makes it more difficult to transfer the approach verbatim to a real-time control system as there will be a significant amount of out-of-distribution EEG signals being fed into the trained model. One intermediary step before moving to a BBMI application could therefore be to re-train these models using the full set of original event labels to see if our conclusions still hold. Future work should ideally focus on studying sparse neural network algorithms on a variety of larger BBMI datasets. Many of our conclusions are a direct consequence of training on a small dataset of twelve participants with a reduced number of class labels, and it is unclear if these conclusions will consistently present in larger, more diverse or contextual datasets. Future work should also consider systematic hyperparameter sensitivity analyses of these algorithms, to understand better how they would perform in real-world scenarios where tuning is impossible.

[2] https://github.com/lpetrich/SparsEEG

VI. Conclusion

In this work we evaluated the performance of two sparse neural network algorithms for an EEG-based upper-limb motion classification task. It is well-known that sparse neural networks can reduce the computational and memory requirements of deep-learning systems [9], [13], and our goal was to determine whether such networks are viable for adaptive BBMI systems. Our results show how sparse neural networks can achieve higher performance and generalization than dense neural networks on one EEG dataset. Furthermore, our experiments provide evidence that sparse neural networks can achieve high accuracy with low variance during learning, making these networks candidates for real-time BBMI systems which adapt to the end-user during deployment. For example, improvements in the performance and stability of EEG-based control for wheelchair-mounted robotic arms could help empower individuals to perform everyday tasks such as eating, drinking, and grasping objects independently. In addition to their use in rehabilitation and assistive robotics, we expect these insights into sparsity to inform the development of other non-medical biomechatronic and bio-inspired systems with limited on-board computational resources. We hope that the present research has therefore laid the groundwork necessary for conducting ambitious future studies into the intuitive and computationally efficient user control of BBMI systems such as robotic arms and other assistive technologies.

Acknowledgment

The authors would like to thank Johannes Günther, Matthew E. Taylor, and Alex Murphy for many helpful discussions and feedback on this paper. Special thanks to Shaylee Lorrain for graphical design assistance.

References

[1] B. Maiseli, A. T. Abdalla, L. V. Massawe, M. Mbise, K. Mkocha, N. A. Nassor, M. Ismail, J. Michael, and S. Kimambo, "Brain–computer interface: Trend, challenges, and threats," *Brain Informatics*, vol. 10, no. 1, p. 20, 2023.

[2] A. N. Belkacem, N. Jamil, J. A. Palmer, S. Ouhbi, and C. Chen, "Brain computer interfaces for improving the quality of life of older adults and elderly patients," *Frontiers in Neuroscience*, vol. 14, p. 692, 2020.

[3] A. Biasiucci, B. Franceschiello, and M. M. Murray, "Electroencephalography," *Current Biology*, vol. 29, no. 3, pp. R80–R85, 2019.

[4] M. S. Al-Quraishi, I. Elamvazuthi, S. A. Daud, S. Parasuraman, and A. Borboni, "EEG-based control for upper and lower limb exoskeletons and prostheses: A systematic review," *Sensors*, vol. 18, no. 10, p. 3342, 2018.

[5] M. Bamdad, H. Zarshenas, and M. A. Auais, "Application of BCI systems in neurorehabilitation: A scoping review," *Disability and Rehabilitation: Assistive Technology*, vol. 10, no. 5, pp. 355–364, 2015.

[6] E. H. T. Shad, M. Molinas, and T. Ytterdal, "Impedance and noise of passive and active dry EEG electrodes: A review," *IEEE Sensors Journal*, vol. 20, no. 24, pp. 14 565–14 577, 2020.

[7] H. Altaheri, G. Muhammad, M. Alsulaiman, S. U. Amin, G. A. Altuwaijri, W. Abdul, M. A. Bencherif, and M. Faisal, "Deep learning techniques for classification of electroencephalogram (EEG) motor imagery (MI) signals: A review," *Neural Computing and Applications*, vol. 35, no. 20, pp. 14 681–14 722, 2023.

[8] F. Lotte, M. Congedo, A. Lécuyer, F. Lamarche, and B. Arnaldi, "A review of classification algorithms for EEG-based brain–computer interfaces," *Journal of Neural Engineering*, vol. 4, no. 2, p. R1, 2007.

[9] M. Lasby, A. Golubeva, U. Evci, M. Nica, and Y. Ioannou, "Dynamic sparse training with structured sparsity," *Proceedings of the 12th International Conference on Learning Representations*, 2024.

[10] D. C. Mocanu, E. Mocanu, P. Stone, P. H. Nguyen, M. Gibescu, and A. Liotta, "Scalable training of artificial neural networks with adaptive sparse connectivity inspired by network science," *Nature Communications*, vol. 9, no. 1, p. 2383, 2018.

[11] G. Bellec, D. Kappel, W. Maass, and R. Legenstein, "Deep rewiring: Training very sparse deep networks," in *Proceedings of the 6th International Conference on Learning Representations*, 2018.

[12] S. Liu, "Learning sparse neural networks for better generalization," in *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pp. 5190–5191, 2020.

[13] T. Hoefler, D. Alistarh, T. Ben-Nun, N. Dryden, and A. Peste, "Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks," *The Journal of Machine Learning Research*, vol. 22, no. 1, pp. 10 882–11 005, 2021.

[14] V. Liu, R. Kumaraswamy, L. Le, and M. White, "The utility of sparse representations for control in reinforcement learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, pp. 4384–4391, 2019

[15] S. Ghiassian, B. Rafiee, Y. L. Lo, and A. White, "Improving performance in reinforcement learning by breaking generalization in neural networks," in *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, 2020.

[16] Y. Pan, K. Banman, and M. White, "Fuzzy tiling activations: A simple approach to learning sparse representations online," in *Proceedings of the 9th International Conference on Learning Representations*, 2021.

[17] B. Grooten, G. Sokar, S. Dohare, E. Mocanu, M. E. Taylor, M. Pechenizkiy, and D. C. Mocanu, "Automatic noise filtering with dynamic sparse training in deep reinforcement learning," in *Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems*, 2023.

[18] A. L. Barth and J. F. Poulet, "Experimental evidence for sparse firing in the neocortex," *Trends in Neurosciences*, vol. 35, no. 6, pp. 345–355, 2012.

[19] A. J. Holtmaat, J. T. Trachtenberg, L. Wilbrecht, G. M. Shepherd, X. Zhang, G. W. Knott, and K. Svoboda, "Transient and persistent dendritic spines in the neocortex in vivo," *Neuron*, vol. 45, no. 2, pp. 279–291, 2005.

[20] D. D. Stettler, H. Yamahachi, W. Li, W. Denk, and C. D. Gilbert, "Axons and synaptic boutons are highly dynamic in adult visual cortex," *Neuron*, vol. 49, no. 6, pp. 877–887, 2006.

[21] A. R. Chambers and S. Rumpel, "A stable brain from unstable components: Emerging concepts and implications for neural computation," *Neuroscience*, vol. 357, pp. 172–184, 2017.

[22] P. Erdős and A. Rényi, "On random graphs I," *Publicationes Mathematicae Debrecen*, vol. 6, no. 18, pp. 290–297, 1959.

[23] S. A. Janowsky, "Pruning versus clipping in neural networks," *Physical Review A*, vol. 39, no. 12, pp. 6600–6603, Jun. 1989.

[24] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural network," *Proceedings of the 28th International Conference on Neural Information Processing Systems*, vol. 28, 2015.

[25] M. Zhu and S. Gupta, "To prune, or not to prune: Exploring the efficacy of pruning for model compression," in *Proceedings of the 6th International Conference on Learning Representations Workshop Track*, 2018.

[26] J. Frankle and M. Carbin, "The lottery ticket hypothesis: Finding sparse, trainable neural networks," in *Proceedings of the 7th International Conference on Learning Representations*, 2019.

[27] M. D. Luciw, E. Jarocka, and B. B. Edin, "Multi-channel EEG recordings during 3,936 grasp and lift trials with varying weight and friction," *Scientific Data*, vol. 1, no. 1, pp. 1–11, 2014.

[28] N. Yahya, H. Musa, Z. Y. Ong, and I. Elamvazuthi, "Classification of motor functions from electroencephalogram (EEG) signals based on an integrated method comprised of common spatial pattern and wavelet transform framework," *Sensors*, vol. 19, no. 22, p. 4878, 2019.

[29] T. Liu and D. Yang, "A three-branch 3D convolutional neural network for EEG-based different hand movement stages classification," *Scientific Reports*, vol. 11, no. 1, p. 10758, 2021.

[30] S. Li and H. Feng, "EEG signal classification method based on feature priority analysis and CNN," in *2019 International Conference on Communications, Information System and Computer Engineering (CISCE)*, pp. 403–406, 2019.

[31] B. Edin, K. Greff, and W. Cukierski, "Grasp-and-lift EEG detection," 2015. [Online]. Available: https://kaggle.com/competitions/grasp-and-lift-eeg-detection

[32] X. Jiang, G.-B. Bian, and Z. Tian, "Removal of artifacts from EEG signals: A review," *Sensors*, vol. 19, no. 5, p. 987, Feb. 2019.

[33] R. Janapati, V. Dalal, and R. Sengupta, "Advances in modern EEG-BCI signal processing: A review," *Materials Today: Proceedings*, vol. 80, pp. 2563–2566, 2023.

[34] J. Heek, A. Levskaya, A. Oliver, M. Ritter, B. Rondepierre, A. Steiner, and M. van Zee, "Flax: A neural network library and ecosystem for JAX," 2023. [Online]. Available: http://github.com/google/flax

[35] J. H. Lee, W. Park, N. Mitchell, J. Pilault, J. S. Obando-Ceron, H.-B. Kim, N. Lee, E. Frantar, Y. Long, A. Yazdanbakhsh, S. Agrawal, S. Subramanian, X. Wang, S.-C. Kao, X. Zhang, T. Gale, A. J. C. Bik, W. Han, M. Ferev, Z. Han, H.-S. Kim, Y. Dauphin, K. Dziugaite, P. S. Castro, and U. Evci, "JaxPruner: A concise library for sparsity research," 2023.

[36] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *Proceedings of the 3rd International Conference on Learning Representations*, 2014.

[37] DeepMind, I. Babuschkin, K. Baumli, A. Bell, S. Bhupatiraju, J. Bruce, P. Buchlovsky, D. Budden, T. Cai, A. Clark, I. Danihelka, A. Dedieu, C. Fantacci, J. Godwin, C. Jones, R. Hemsley, T. Hennigan, M. Hessel, S. Hou, S. Kapturowski, T. Keck, I. Kemaev, M. King, M. Kunesch, L. Martens, H. Merzic, V. Mikulik, T. Norman, G. Papamakarios, J. Quan, R. Ring, F. Ruiz, A. Sanchez, L. Sartran, R. Schneider, E. Sezener, S. Spencer, S. Srinivasan, M. Stanojević, W. Stokowiec, L. Wang, G. Zhou, and F. Viola, "The DeepMind JAX ecosystem," 2020. [Online]. Available: http://github.com/google-deepmind

[38] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and Édouard Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, no. 85, pp. 2825–2830, 2011.

[39] R. Novak, Y. Bahri, D. A. Abolafia, J. Pennington, and J. Sohl-Dickstein, "Sensitivity and generalization in neural networks: An empirical study," in *Proceedings of the 6th International Conference on Learning Representations*, 2018.

[40] S. Liu, D. C. Mocanu, and M. Pechenizkiy, "On improving deep learning generalization with adaptive sparse connectivity," *ICML 2019 Workshop on Understanding and Improving Generalization in Deep Learning*, 2019.

[41] S. Saha and M. Baumert, "Intra- and inter-subject variability in EEG-based sensorimotor brain computer interface: A review," *Frontiers in Computational Neuroscience*, vol. 13, p. 87, 2020.

[42] F. Fahimi, Z. Zhang, W. B. Goh, T.-S. Lee, K. K. Ang, and C. Guan, "Inter-subject transfer learning with an end-to-end deep convolutional neural network for EEG-based BCI," *Journal of Neural Engineering*, vol. 16, no. 2, p. 026007, Jan. 2019.

[43] H. Suresh and J. Guttag, "A framework for understanding sources of harm throughout the machine learning life cycle," in *Equity and Access in Algorithms, Mechanisms, and Optimization*, pp. 1–9, Oct. 2021.